TUGAS AKHIR – KS 141501

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

DEVELOPMENT OF PRODUCT CATALOG'S METADATA MANAGEMENT FEATURE FOR SOFTWARE SALES SITES

FAHRIZAL ADI WIBOWO
NRP 5213 100 173

Dosen Pembimbing :
Rully Agus Hendrawan, S.Kom., M.Eng.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman ini sengaja dikosongkan

TUGAS AKHIR – KS 141501

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

FAHRIZAL ADI WIBOWO
NRP 5213 100 173

Dosen Pembimbing :
Rully Agus Hendrawan, S.Kom., M.Eng.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman ini sengaja dikosongkan

FINAL PROJECT – KS 141501

DEVELOPMENT OF PRODUCT CATALOG'S METADATA MANAGEMENT FEATURE FOR SOFTWARE SALES SITES

FAHRIZAL ADI WIBOWO
NRP 5213 100 173

Supervisor:
Rully Agus Hendrawan, S.Kom., M.Eng.

INFORMATION SYSTEMS DEPARTMENT
Information Technology Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

FAHRIZAL ADI WIBOWO
5213 100 173

Surabaya, 04 Juli 2017

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M. Kom.
NIP 19650310199102001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Oleh :

FAHRIZAL ADI WIBOWO
NRP. 5213 100 173

Disetujui Tim Penguji : Tanggal Ujian: 04 Juli 2017
Periode Wisuda: September 2017

Rully Agus Hendrawan, S.Kom., M.Eng.	**(Pembimbing I)**
Mahendrawati ER, ST., M.Sc., Ph.D.	**(Penguji I)**
Arif Wibisono, S.Kom, M.Sc.	**(Penguji II)**

Halaman ini sengaja dikosongkan

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

Nama Mahasiswa : Fahrizal Adi Wibowo
NRP : 5213 100 173
Departemen : Sistem Informasi FTIf-ITS
Pembimbing I : Rully Agus Hendrawan, S.Kom., M.Eng.

ABSTRAK

Pada tahun 2016, Apple App Store yang telah memiliki lebih dari 2.2 juta aplikasi dan Google Play Store memiliki lebih dari 2 juta aplikasi pada Direktori Aplikasi mereka. Dimana hal tersebut semakin mendorong lahirnya berbagai Direktori Aplikasi dari berbagai Perusahaan Teknologi sesuai dengan platform yang mereka miliki seperti Apple App Store, Google Play Store, Windows Store dll.

Hal tersebut memberikan tantangan lebih bagi Pengembang Direktori Aplikasi untuk Menyediakan maupun Mengelola Data Produk Perangkat Lunak yang terus tumbuh dan berkembang mengikuti perkembangan teknologi yang ada dan juga bagi Perusahaan Mesin Pencari (Search Engine) untuk menghasilkan pencarian yang relevan bagi penggunanya.

Tugas Akhir ini merancang Database Schema yang memungkinkan untuk dapat selalu di rubah secara dinamis dan juga memungkinkan adanya Schema & Data Versioning untuk merekam perubahan yang terjadi pada Schema & Data. Rancangan ini dibangun melalui Teknologi Laravel PHP Framework dan WYSIWYG yang bertujuan untuk memfasilitasi Pengembang dalam Mengelola Katalog Penjualan Produk Perangkat Lunak dan membantu Perusahaan Mesin Pencari dalam menginterpretasikan informasi pada sebuah halaman dengan mudah untuk dapat digunakan meningkatkan hasil

pencarian bagi Pengguna. Sehingga, nantinya Pengembang tidak perlu lagi melakukan Integrasi / Migrasi Basis Data maupun Kode Program untuk mengubah Struktur Data Katalog Produk Perangkat Lunak dan juga Perusahaan Mesin Pencari dapat menghasilkan pencarian yang jauh lebih baik dan relevan bagi penggunanya dengan kemudahan dalam menginterpretasikan informasi pada sebuah halaman produk yang telah terintegrasi dengan Schema.org.

Kata kunci: Metadata, Schema, Pengelolaan

DEVELOPMENT OF PRODUCT CATALOG'S METADATA MANAGEMENT FEATURE FOR SOFTWARE SALES SITES

Student Name : Fahrizal Adi Wibowo
NRP : 5213 100 173
Department : Sistem Informasi FTIf-ITS
Supervisor I : Rully Agus Hendrawan, S.Kom., M.Eng

ABSTRACT

By 2016, the Apple App Store that already has more than 2.2 million apps and the Google Play Store has over 2 million apps in their App Directory. Where it is increasingly encouraging the birth of various Applications Directories from various Technology Companies in accordance with their platforms such as Apple App Store, Google Play Store, Windows Store etc.

It poses more challenges for Application Directory Developers to Provide or Manage Software Product Data that continues to grow and evolve as technology develops as well as for Search Engines to generate relevant searches for its users.

This Final Project designed a Schema Database that allows for dynamic alignment and also allows for Schema & Data Versioning to record changes that occur in Schema & Data. The design is built through Laravel PHP Framework and WYSIWYG Technology which aims to facilitate Developers in Managing the Catalog of Product Sales of the Software and assist Search Engines Companies in interpreting information on a page with ease to be used to improve search results for Users. Thus, developers will no longer need to integrate / migrate the Database or Program Code to change the Product Data Catalog Structure of the Software as well as the Search Engines Company to produce a much better and relevant search for its users with ease in interpreting information on a product page that Has been integrated with Schema.org.

Keywords: ***Metadata, Schema, Management***

KATA PENGANTAR

Puji syukur kepada Tuhan yang Maha Esa sehingga penulis dapat menyelesaikan buku tugas akhir dengan judul:

PENGEMBANGAN FITUR PENGELOLAAN METADATA KATALOG PRODUK PADA SITUS PENJUALAN PERANGKAT LUNAK

yang merupakan salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pengerjaan tugas akhir yang berlangsung selama satu semester, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada yang senantiasa terlibat secara langsung memberikan bantuan dan dukungan dalam pengerjaan tugas akhir ini:

- Tuhan yang Maha Esa yang telah memberikan kesehatan, kemudahan, kekuatan, kelancaran dan kesempatan untuk penulis hingga dapat menyelesaikan Tugas Akhir ini.
- Kedua orang tua, dan keluarga yang selalu hadir senantiasa dengan doa dan dukungan untuk menyelesaikan Tugas Akhir ini.
- Bapak Dr. Ir. Aris Tjahyanto, M.Kom, selaku Kepala Departemen Sistem Informasi ITS, yang telah menyediakan fasilitas terbaik untuk kebutuhan penelitian mahasiswa.
- Bapak Rully Agus Hendrawan, S.Kom., M.Eng selaku dosen pembimbing yang yang telah banyak meluangkan waktu untuk membimbing, mengarahkan, dan mendukung dalam penyelesaian Tugas Akhir.
- Ibu Nur Aini Rakhmawati S.Kom, M.Sc.Eng selaku dosen wali yang telah memberikan arahan terkait perkuliahan di Departemen Sistem Informasi.
- Seluruh dosen pengajar beserta staff dan karyawan di Departemen Sistem Informasi, FTIf ITS Surabaya yang

telah memberikan ilmu dan bantuan kepada penulis selama 8 semester ini.

- Teman-teman seperjuangan Angkatan 2013 Beltranis, yang selalu memberikan semangat positif untuk menyelesaikan Tugas Akhir dengan tepat waktu.

Penulis menyadari bahwa Tugas Akhir ini masih belum sempurna dan memiliki banyak kekurangan di dalamnya. Dan oleh karena itu, penulis meminta maaf atas segala kesalahan yang dibuat penulis dalam buku Tugas Akhir ini. Penulis membuka pintu selebar-lebarnya bagi pihak yang ingin memberikan kritik dan saran, dan penelitian selanjutnya yang ingin menyempurnakan karya dari Tugas Akhir ini. Semoga buku Tugas Akhir ini bermanfaat bagi seluruh pembaca.

Surabaya, 04 Juli 2017

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab pendahuluan akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan tugas akhir, manfaat kegiatan tugas akhir dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir dapat dipahami.

1.1. Latar Belakang Masalah

Mobile Application Market telah mengalami lonjakan pertumbuhan dalam beberapa tahun terakhir terutama pada tahun 2016, dimana Apple App Store yang telah memiliki lebih dari 2.2 juta aplikasi pada Direktori Aplikasi mereka dan Google Play Store mengikuti dengan selisih yang tidak terlalu jauh dari Apple App Store kini memiliki lebih dari 2 juta aplikasi pada Direktori Aplikasi mereka [1].

Dengan semakin bertumbuhnya perkembangan *Mobile Application* yang ada di pasar saat ini, hal tersebut juga di ikuti dengan semakin berkembangnya berbagai macam Direktori Aplikasi dimana pada saat ini Pengguna semakin menuntut tidak hanya agar Aplikasi tersebut dapat berjalan dengan baik dan semestinya, tetapi juga dapat berjalan pada berbagai *platform* berbeda seperti iPhone, Android, Windows dll [2]. Dimana hal tersebut yang semakin mendorong lahirnya berbagai Direktori Aplikasi dari berbagai Perusahaan Teknologi sesuai dengan *platform* yang dimilikinya seperti Apple App Store, Google Play Store, Windows Store dll.

Hal tersebut dapat memberikan tantangan tersendiri bagi Pengembang Direktori Aplikasi terkait bagaimana menyediakan Direktori Aplikasi yang dapat memenuhi kebutuhan Penggunanya untuk menemukan aplikasi yang mereka butuhkan dan sekaligus memfasilitasi kebutuhan

Pengembang Direktori Aplikasi (*Database Designer* dan *Developer*) dalam Mengelola Data Produk Perangkat Lunak pada Direktori Aplikasi yang terus tumbuh dan berkembang dengan mudah dan dinamis. Tantangan ini juga akan saling terkait dengan Perusahaan Mesin Pencari (*Search Engine*) untuk menghasilkan pencarian yang relevan bagi pengguna untuk menemukan Aplikasi yang dibutuhkan.

Oleh karena itu, dibutuhkan Skema Basis Data yang secara dinamis dapat menampung dan mengelola Data Produk Perangkat Lunak pada Direktori Aplikasi tersebut dengan mudah dan cepat. Sebuah Skema Basis Data / *Database Schema* merupakan sebuah struktur kerangka yang mendefinisikan bagaimana data diatur dan bagaimana keterkaitan hubungan di antara mereka [3]. Pada *Software Application*, skema ini sebelumnya telah didefinisikan oleh Schema.org dimana skema ini dinamis dan dapat dipakai dan di sesuaikan pada berbagai bentuk *platform* Produk Perangkat Lunak yang juga terus berkembang seiring bertambahkan *platform* yang hadir pada industri teknologi [4]. Skema tersebut bukan ditujukan langsung untuk Pengguna, tetapi lebih di utamakan bagi *Developer* dalam melakukan integrasi aplikasi korporasi dan mesin pencari untuk menemukan informasi Produk Perangkat Lunak yang relevan bagi penggunanya.

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang, maka rumusan permasalahan yang menjadi fokus dan akan diselesaikan dalam Tugas Akhir ini antara lain:

1. Bagaimana mengelola Struktur Data Katalog Produk Perangkat Lunak secara mudah sesuai kebutuhan yang dinamis?
2. Bagaimana mengubah Struktur Data Katalog Produk Perangkat Lunak tanpa harus melakukan Migrasi dan Integrasi Basis Data?

3. Bagaimana mengelola Struktur Data Katalog Produk Perangkat Lunak tanpa harus mengubah Kode Program secara Langsung?

1.3. Batasan Masalah

Dari permasalahan yang disebutkan di atas, batasan masalah dalam tugas akhir ini adalah :

1. Aplikasi merupakan berbasis Web yang dibangun dengan *Laravel PHP Framework* dan JavaScript.
2. RDBMS menggunakan *MySQL Database*.
3. Sumber Data Produk bertipe Perangkat Lunak berasal dari Google Play Store.

1.4. Tujuan Tugas Akhir

Berdasarkan hasil perumusan masalah dan batasan masalah yang telah disebutkan sebelumnya, maka tujuan yang dicapai dari tugas akhir ini adalah merancang dan membangun Aplikasi Web Berbasis *Laravel PHP Framework* Pengelolaan Metadata pada Katalog Penjualan Produk Perangkat Lunak. Dimana setiap Produk yang tampil di Web dapat terbaca dalam format *Semantic Web* terstruktur mengikuti standar Schema.org.

1.5. Manfaat Tugas Akhir

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah:

1. Bagi *Developer* situs *eCommerce* (Direktori Aplikasi / *Software Directory*) yaitu Meningkatkan Kemampuan Struktur Data yang Dinamis ketika melakukan Pengembangan Sistem dan Konten dan juga Mempermudah *Database Designer* dan *Developer* dalam mengelola Struktur Data Produk.
2. Bagi Pemilik situs *eCommerce* (Direktori Aplikasi / *Software Directory*) yaitu Mengurangi Tingkat Kesulitan

dan Lamanya Proses Perubahan dan Pengelolaan Struktur Data Produk.

3. Bagi Perusahaan *Search Engine* yaitu Menyediakan Data yang Terstruktur (*Semantic Web*) agar dapat dibaca dengan mudah oleh *Search Engine*.

1.6. Relevansi

Tugas akhir ini berkaitan dengan mata kuliah Interaksi Manusia Komputer, Algoritma dan Pemrograman, Pemrograman Berorientasi Objek, Pemrograman Berbasis Web, Desain Basis Data, Analisa dan Desain Perangkat Lunak, Pemrograman Integratif dan Konstruksi Pengembangan Perangkat Lun

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan mengenai penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini. Landasan teori akan memberikan gambaran secara umum dari landasan penjabaran tugas akhir ini.

2.1. Penelitian Sebelumnya

Penelitian yang dijadikan acuan dalam pengerjaan tugas akhir ini diantaranya sebagai berikut :

2.1.1. Penelitian Terkait 1

Tabel 2.1 Penelitian Terkait 1

Judul Patent	*Flexible database schema* [5]
Metode	Berbagai metode dapat dilakukan untuk menciptakan sebuah skema database fleksibel. Pada penelitian ini, metode yang digunakan yaitu dengan menyimpan informasi identifikasi entitas dalam sebuah tabel. Dimana pada tabel lain terdapat tabel yang menyimpan informasi yang menggambarkan entitas berikut metadata yang dimiliki dan relasi dari entitas yang ada.
Penemu Patent	Craig Robert King, Richard K. Freeman
Tanggal Penerbitan	12 Mei 2015

Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi untuk mengelola dan merancang basis data yang akan digunakan oleh Katalog Produk untuk mengurangi susahnya dan lamanya perubahan dan pengelolaan struktur data produk.

2.1.2. Penelitian Terkait 2

Tabel 2.2 Penelitian Terkait 2

Judul Patent	*Integrated collaborative user interface for a document editor program* [6]
Metode	Penelitian ini merancang Antarmuka Pengguna pada *Document Editor*. Antarmuka pengguna ini dirancang agar kolaboratif dan terintegrasi mencakup panel dokumen, rincian dokumen dan rincian bagian. Berikut menyediakan *update* dokumen dalam *Document Editor* terkait perubahan, penambahan dan penghapusan *(Versioning)*. Setiap perubahan tersebut akan dilakukan *refresh* halaman untuk memberikan konsistensi dan status terbaru dari dokumen.
Penemu Patent	Jared R. Parker, Sangya Singh, Greg Prickril, Wai Chan
Tanggal Penerbitan	1 Jan 2009

Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi antarmuka dan interaksi yang dapat dilakukan dalam mengelola metadata pada halaman Katalog Produk terkait Informasi dan Rincian dari Produk Perangkat Lunak.

2.1.3. Penelitian Terkait 3

Tabel 2.3 Penelitian Terkait 3

Judul Patent	*Self-describing editors for browser-based WYSIWYG XML/HTML editors* [7]
Metode	Penelitian ini merancang sebuah metode dan sistem untuk dapat mengubah format *XML Web Content* sesuai kebutuhan melalui sebuah antarmuka pengguna grafis (*GUI*) yang di khusus agar dapat berjalan pada sebuah *Browser* yang berjalan pada sisi *client*. Karena XML sangat tidak ramah untuk dapat dibaca atau diedit oleh manusia, maka *XML Web Content* tersebut diubah dalam *web-mark-up* yang lebih mudah dibaca dan dimodifikasi. Proses perubahan dilakukan pada *Browser* dengan menampilkan konten XML untuk dapat dilakukan perubahan secara langsung pada halaman itu juga.
Penemu Patent	Andreas Seurig, Thomas Spillecke

Tanggal Penerbitan	29 Jun 2006
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi dalam merancang Katalog Produk menjadi sebuah Data Terstruktur (*Semantic Web*) yang memungkinkan untuk dibaca dengan baik oleh Mesin Pencari / *Search Engine*.

2.1.4. Penelitian Terkait 4

Tabel 2.4 Penelitian Terkait 4

Judul Patent	*Method and system for designing, editing and publishing web page content in a live internet session* [8]
Metode	Penelitian ini membahas sebuah metode dan sistem untuk merancang, mengubah dan mempublikasi sebuah konten halaman web pada *live Internet session* untuk menyajikan desain halaman penerbitan konten web secara langsung secara berkualitas dan profesional. Server menyediakan sesi *editing* untuk menerima informasi dari interaksi antarmuka pengguna untuk mendeteksi perubahan konten web dan menyajikan konten halaman web yang sebenarnya secara langsung pada saat itu juga pada antarmuka pengguna.

Penemu Patent	Dennis Altshuler
Tanggal Penerbitan	28 Okt 2004
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi antarmuka dan interaksi terhadap perubahan metadata Katalog Produk dengan menampilkan hasil perubahannya secara langsung pada halaman web saat itu juga tanpa harus memuat ulang halaman web.

2.1.5. Penelitian Terkait 5

Tabel 2.5 Penelitian Terkait 5

Judul Patent	*Method and apparatus for generating and modifying multiple instances of element of a web site* [9]
Metode	Penelitian ini membahas tentang sebuah metode yang digunakan untuk memodifikasi dan menghasilkan kerangka pada sebuah situs web berdasarkan karakteristik dan sesuai dengan entri data yang telah di tentukan sebelumnya. Metode ini memungkinkan untuk menghasilkan kerangka sebuah halaman multi-dimensi dari situs web berdasarkan karakteristik yang telah ditentukan dan melakukan regenerasi kerangka

	berdasarkan karakteristik yang telah dimodifikasi / diperbarui.
Penemu Patent	John Underwood Paul Neilson Hanson Char David Shing Peter Horner Mark Underwood Darren Slaney Gary Evesson
Tanggal Penerbitan	24 Feb 2004
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi untuk membuat dan mengelola kerangka atau struktur metadata yang nantinya akan digunakan sebagai acuan pada sebuah Informasi dalam Produk Perangkat Lunak.

2.1.6. Penelitian Terkait 6

Tabel 2.6 Penelitian Terkait 6

Judul Penelitian	*The web page as a WYSIWYG end-user customizable database-backed information management application* [10]
Metode	Pada penelitian ini, metode yang digunakan yaitu dengan memanfaatkan AJAX sebagai *visual editor* interaktif untuk mengelola sebuah halaman web yang berisikan data terstruktur dan kaya. Dan juga dengan memanfaatkan WYSIWYG sebagai *MetaEditor*. Metode ini memungkinkan pengguna untuk mengelola atau mengubah secara

	langsung konten yang kaya dalam sebuah halaman web.
Penulis	Karger, David R., Scott Ostler, and Ryan Lee
Tahun Penerbitan	2009
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi interaksi dan antarmuka dalam mengelola konten yang begitu kaya dalam sebuah halaman web dimana perubahan dapat dilakukan berikut hasilnya dapat dilihat secara langsung pada saat itu juga dalam halaman web.

2.1.7. Penelitian Terkait 7

Tabel 2.7 Penelitian Terkait 7

Judul Penelitian	*Mavo: Creating Interactive Data-Driven Web Applications by Authoring HTML* [11]
Metode	Penelitian ini membahas sebuah metode untuk mengubah sebuah halaman web statis menjadi sebuah aplikasi web interaktif, dimana dapat memungkinkan pengguna untuk mendefinisikan skema data yang kompleks yang dirancang dalam sebuah halaman HTML dengan menambahkan beberapa *attributes* dan *expressions* pada HTML *elements* untuk mengubah halaman web

	statis tersebut menjadi *data-driven web application* dimana data yang ada dalam halaman tersebut dapat dirubah atau dimanipulasi secara langsung melalui *browser*.
Penulis	Verou, Lea, Amy X. Zhang, and David R. Karger
Tahun Penerbitan	2016
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi dalam mengelola data dan metadata dalam sebuah halaman web dimana perubahan dan manipulasi dapat dilakukan secara langsung melalui bantuan *browser* tanpa harus membuat pengguna mengubah kode program dalam halaman web tersebut dengan susah payah.

2.1.8. Penelitian Terkait 8

Tabel 2.8 Penelitian Terkait 8

Judul Penelitian	*The Callimachus project: RDFa as a web template language* [12]
Metode	Penelitian ini membahas sebuah metode dengan memanfaatkan RDFa sebagai bahasa template yang memungkinkan untuk mengolah sebuah halaman web yang biasa dibaca oleh manusia, dapat juga dibaca dengan mudah oleh mesin dengan cara memadukan teknologi XHTML

	dengan RDFa pada sebuah halaman web.
Penulis	S. Battle, D. Wood, J. Leigh, L. Ruth
Tahun Penerbitan	2012
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi dalam merancang sebuah halaman Katalog Produk yang tidak hanya dapat dibaca dengan mudah oleh penggunanya tetapi juga dapat dibaca dengan mudah juga oleh Mesin Pencari / *Search Engine.*

2.1.9. Penelitian Terkait 9

Tabel 2.9 Penelitian Terkait 9

Judul Penelitian	*Dabblet: A visual IDE for rapid prototyping of client-side web development* [13]
Metode	Penelitian ini membahas sebuah metode dan sistem untuk merancang sebuah web yang memungkinkan pengguna untuk mengelola atau mengubah konten dalam sebuah halaman secara *real-time*. Metode ini lebih dominan kepada perancangan pengalaman pengguna yang interaktif dalam mengelola atau berkolaborasi pada sebuah halaman web.

Penulis	Michailia Verou
Tahun Penerbitan	2013
Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi dalam menciptakan sebuah lingkungan atau pengalaman pengguna yang interaktif dan kolaboratif dalam sebuah halaman web.

2.1.10. Penelitian Terkait 10

Tabel 2.10 Penelitian Terkait 10

Judul Penelitian	*Do-It-Yourself database-driven web applications* [14]
Metode	Penelitian ini membahas sebuah metode dan sistem untuk merancang sebuah formulir dalam halaman web berikut metadata pada formulir tersebut secara hierarki dan terstruktur dengan memanfaatkan berbagai kerangka yang sudah disediakan sebelumnya dimana dapat digunakan secara leluasa dan dengan begitu mudah pada halaman formulir tersebut.
Penulis	Kowalzcykowski, Keith, Alin Deutsch, Kian Win Ong, Yannis Papakonstantinou, Kevin Keliang Zhao, and Michalis Petropoulos
Tahun Penerbitan	2009

Keterkaitan Tugas Akhir	Metode ini dapat digunakan sebagai referensi dalam merancang kerangka atau *element-element* terkait media apa saja dan model data seperti apa saja yang nantinya dapat digunakan dan bagaimana cara mengelola dan memperlakukannya dalam sebuah halaman web yang interaktif.

2.2. Dasar Teori

2.2.1. Metadata

Metadata secara umum didefinisikan sebagai data terstruktur dari sebuah data dan informasi [15]. Dalam sudut pandang basis data, Metadata adalah basis data yang berisi informasi tentang struktur data, arti data, penggunaan data, aturan kualitas data, dan informasi lain tentang data [16].

Metadata dibutuhkan untuk menggambarkan dan menjelaskan data dan gudang data / *data warehouse* dalam artian struktur data dan proses bagi pengguna. Definisi Metadata dilakukan untuk menghindari kesalah pahaman akan makna dari suatu kolom tertentu dimana pemetaan metadata juga membantu pengguna untuk memahami dampak dari perubahan atau penambahan kolom baru pada basis data.

2.2.2. Schema.org

Schema.org merupakan sebuah inisiatif bersama dari Microsoft, Google, Yahoo! dan Yandex untuk meningkatkan web dengan membuat kosakata umum atau populer untuk mendeskripsikan data di web [17].

Tujuan dari inisiatif ini adalah untuk membantu mesin pencari untuk menginterpretasikan informasi pada halaman web

sehingga dapat digunakan untuk meningkatkan tampilan hasil pencarian, sehingga lebih mudah bagi orang untuk menemukan informasi yang mereka cari. Schema.org memiliki dua komponen diantaranya [18] :

1. *Ontology* yaitu kosakata untuk penamaan jenis dan karakteristik sumber daya dan hubungan mereka satu sama lain dan juga tentang bagaimana untuk menggambarkan karakteristik ini dengan hubungannya.
2. Ekspresi dari informasi dalam format yang dapat dibaca mesin seperti Microdata, RDFa Lite dan JSON-LD.

Penambahan markup Schema.org ke dalam sebuah laman HTML memungkinkan berbagai mesin pencari untuk memahami makna teks dari data atau informasi dalam sebuah halaman web.

2.2.3. *Database Schema*

Database Schema merupakan sebuah struktur kerangka yang mewakili pandangan logis dari basis data. *Database Schema* mendefinisikan bagaimana data diatur dan bagaimana keterkaitan hubungan di antara mereka [3]. *Database Schema* memberikan deskripsi hubungan logis secara lengkap dari basis data yang meliputi rincian nama-nama dari tiap *field* (atau atribut atau kolom) dan tipe dari tiap *field*.

Pada DBMS, *Database Schema* menjelaskan struktur *formal language* yang ada pada DBMS dan dapat dikatakan sebagai penjelasan dari *database* itu sendiri. *Database Schema* menunjukkan bagaimana entitas yang ada dalam *database* berhubungan satu sama lain, termasuk *tables*, *views*, *stored procedures*, dan banyak lagi. *Database Schema* tidak boleh sering berubah karena akan berpengaruh terhadap data yang disimpan di dalam *database* [19].

2.2.4. *Schema & Data Versioning*

Schema Evolution & Versioning merupakan kemampuan untuk mengubah skema dari basis data tanpa kehilangan data (menyediakan akses ke data lama dan baru). *Schema Versioning* memiliki kemampuan untuk mengakses semua data (baik lama dan baru) melalui antarmuka versi yang berbeda [20].

Pada *Schema Evolution* terdapat sebuah isu dimana beberapa versi dari *Ontology* yang sama dapat terikat untuk tetap ada dan saling mendukung. Karena idealnya, pengembang harus menjaga tidak hanya versi yang berbeda dari *Ontology*, tetapi juga beberapa informasi tentang bagaimana versi berbeda dapat *Compatible* antara satu sama lain [20].

Oleh karena itu manajemen perubahan adalah isu utama dalam dukungan pada *Schema Evolution*. Oleh karena itu, Tugas Akhir ini akan menggabungkan *Schema Evolution & Versioning* menjadi konsep tunggal yang didefinisikan sebagai kemampuan untuk mengelola perubahan *Ontology* dan efek mereka dengan menciptakan dan memelihara berbagai varian *Ontology*. Kemampuan ini terdiri dari metode untuk membedakan dan mengenali versi, spesifikasi hubungan antara versi, perubahan dan prosedur perubahan *Ontology* dan mekanisme akses yang menggabungkan berbagai versi ontologi dan data yang sesuai [20].

Untuk mendukung *Data Versioning*, sistem data perlu mempertimbangkan bagaimana untuk menyimpan, menampilkan, dan melakukan ekstraksi versi pada data. Tantangan utama dalam menyimpan data adalah untuk menjaga konsistensi dari data sementara memungkinkan nilai dari data untuk dapat selalu di ubah dengan mudah. Sebuah versi dapat direpresentasikan sebagai perbedaan dari data yang ada dengan data lainnya. *Data Versioning* tidak akan benar-benar berguna kecuali dapat diekstraksi dengan mudah dan cepat. Tidak selalu menjadi hal yang mudah bagi pengguna untuk mengingat versi mana yang ingin digunakan saat merubah data. Oleh karena itu,

Data Versioning tidak akan berarti jika justru membutuhkan waktu lebih lama untuk mengekstrak versi yang ada sebelumnya daripada membuatnya dari awal [21].

2.2.5. WYSIWYG

WYSIWYG merupakan singkatan dari "*what you see is what you get*". Istilah ini digunakan untuk menggambarkan suatu sistem di mana konten yang sedang disunting akan terlihat sama persis dengan hasil keluaran akhirnya jika dalam Tugas Akhir ini akan terkait dengan hasil akhir dari sebuah halaman web karena pada umumnya sebuah pengolah kata biasanya akan terlihat mengerikan dibandingkan dengan hasil akhirnya karena berisi barisan kode yang kurang begitu humanis [22].

Antarmuka WYSIWYG memanfaatkan akses URL dari pengguna sebagai masukan berupa format HTML yang telah di definisikan kedalam lembar kerja dimana dokumen direkonstruksi sedemikian rupa agar setiap potongan teks tertutup dalam tag baru yang dihasilkan oleh parser dengan nilai ID yang sesuai untuk memungkinkan perubahan dokumen tanpa merusak struktur HTML yang telah didefinisikan sebelumnya [23].

2.2.6. *Laravel PHP Framework*

PHP Frameworks memang bukanlah sebuah hal yang baru pada industri pemrograman web terutama, namun salah satu yang terbaru dan populer yaitu Laravel. Laravel telah meledak menjadi salah satu *PHP Frameworks* yang paling populer dan banyak digunakan dalam kurun waktu singkat. Pada saat penulisan, repositori Laravel pada GitHub memiliki lebih banyak bintang daripada *PHP Frameworks* lain yang jauh lebih matang seperti Symfony, CakePHP, CodeIgniter, dan Yii [24].

Terdapat banyak sekali fitur yang dapat meningkatkan produktifitas pengembangan perangkat lunak, namun akan di

bahas beberapa yang akan banyak di eksplorasi pada Tugas Akhir ini diantaranya sebagai berikut [24] :

1. *Modularity*: Laravel dibangun di atas lebih dari 20 *library* yang berbeda yang dimana dibagi menjadi modul-modul secara individu dan terintegrasi dengan *Composer dependency manager* yang dapat memungkinkan penambahan dan pembaruhan dengan sangat mudah.
2. *Configuration management*: Kerap terjadi apabila aplikasi berjalan di lingkungan yang berbeda, yang berarti bahwa pengaturan *database* atau *server e-mail credential* atau yang menampilkan pesan kesalahan akan berbeda perlakuan. Laravel memiliki pendekatan yang konsisten untuk menangani pengaturan konfigurasi, dan pengaturan yang berbeda dapat diterapkan dalam lingkungan yang berbeda melalui penggunaan file .env, yang berisi pengaturan unik untuk lingkungan itu.

Template Engine: Laravel memiliki Blade yang merupakan sebuah bahasa template ringan yang memungkinkan untuk membuat layout hirarkis dengan blok yang telah ditetapkan di mana konten dinamis akan digunakan.

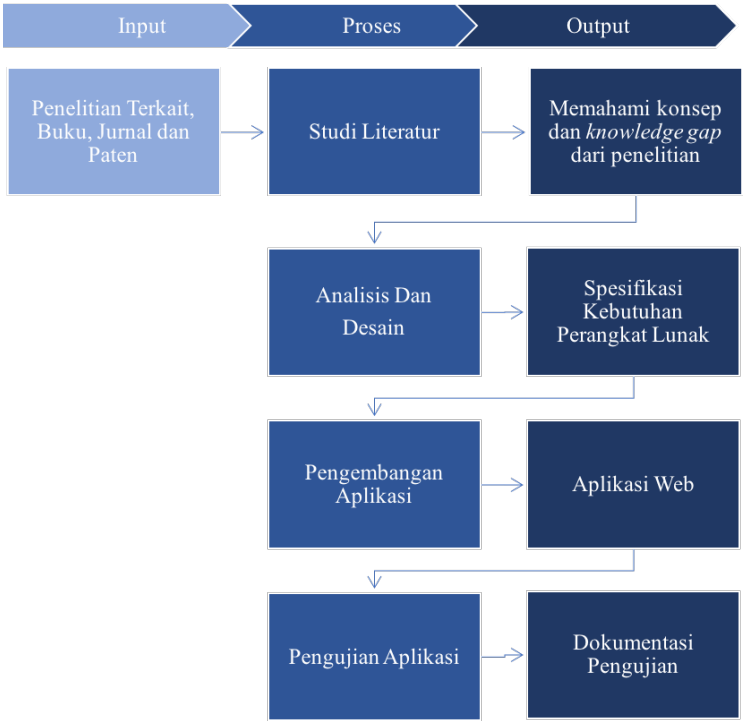
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Pada bab metode penelitian akan dijelaskan mengenai tahapan – tahapan apa saja yang dilakukan dalam pengerjaan tugas akhir ini beserta deskripsi dan penjelasan tiap tahapan tersebut.

3.1. Tahap Pelaksanaan

Pada sub bab ini akan menjelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi ini dapat dilihat pada Gambar 3.1 Metodologi.



Gambar 3.1 Metodologi

3.1.1. Studi Literatur

Pada tahap ini dilakukan pengumpulan literatur yang mendukung dalam menyelesaikan tugas akhir ini. Literatur disini merupakan bentuk dari penjelasan konsep - konsep atau ulasan mengenai penelitian terkait yang pernah dilakukan dan didokumentasikan baik dalam bentuk buku, jurnal, paten maupun website. Luaran dari proses ini adalah pemahaman mengenai konsep dan *knowledge gap* pada penelitian yang pernah ada sebelumnya.

3.1.2. Analisis Dan Desain Aplikasi

Pada tahap ini dilakukan analisa dan desain aplikasi. Setelah mengetahui konsep dan penelitian sebelumnya maka dapat melakukan analisa dan desain aplikasi yang akan dibuat. Pada bagian Analisis, yang harus dilakukan adalah mendaftar Spesifikasi Kebutuhan Perangkat Lunak yang akan dibuat berdasarkan Studi Literatur yang telah dilakukan sebelumya karena Tugas Akhir ini berbasis Kebutuhan Teknologi Informasi maka Daftar Kebutuhan Perangkat Lunak didapatkan dari Studi Literatur dan juga *knowledge gap* yang di dapatkan dari penelitian yang pernah ada atau terkait sebelumnya.

Pada bagian Desain, kebutuhan tersebut akan diterjemahkan kedalam sebuah Perancangan Perangkat Lunak yang berfokus pada Struktur Data, Arsitektur Perangkat Lunak, Representasi Tampilan dan lainnya. Dimana pada tahapan ini akan menghasilkan dokumen yang disebut *Software Requirment* atau Kebutuhan Sistem yang selanjutkan akan digunakan sebagai acuan untuk melakukan Pengembangan Aplikasi.

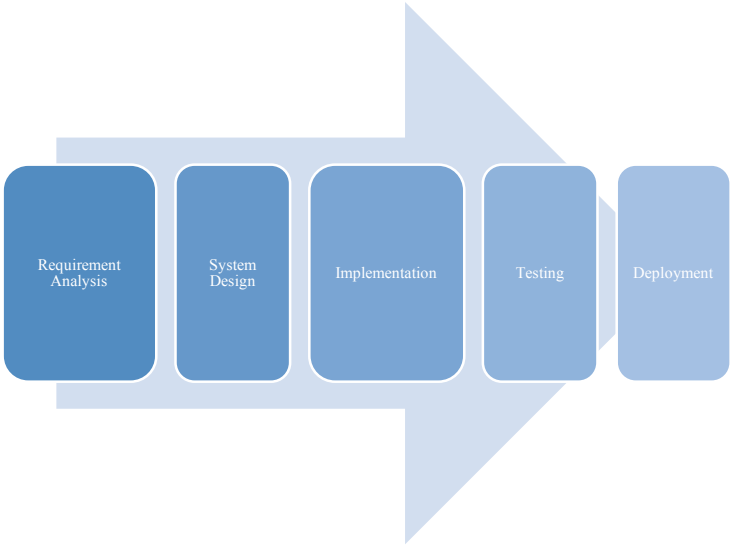
3.1.3. Pengembangan Aplikasi

Pada tahap ini dilakukan pengembangan aplikasi yang merupakan implementasi dari hasil melakukan analisa dan desain aplikasi. Aplikasi dikembangkan berdasarkan

fungsionalitas yang telah didaftar dan arsitektur pada sistem yang telah didesain menggunakan metode *Waterfall*.

3.1.3.1. *Waterfall*

Metodologi *Waterfall* merupakan sebuah metode pengembangan perangkat lunak yang bersifat sekuensial dimana model ini merupakan model yang paling banyak dipakai oleh para pengembang software. Metode *Waterfall* melibatkan serangkaian langkah secara berurutan dimana kemajuan dipandang sebagai aliran yang terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi) dan pengujian [25].



Gambar 3.2 Alur Metode Waterfall

Alur Metode *Waterfall* seperti tampak pada Gambar 3.2 Alur Metode Waterfall dimulai dengan *Requirement Analysis* yang bertujuan untuk memahami perangkat lunak yang diharapkan dan batasan perangkat lunak tersebut, yang dapat didapatkan dari melakukan sebuah penelitian, wawancara ataupun studi literatur. Dimana pada Tugas Akhir ini Spesifikasi Kebutuhan Perangkat Lunak didapatkan dari Studi Literatur yang telah

dilakukan sebelumya dan juga *knowledge gap* yang di dapatkan dari penelitian yang pernah ada atau terkait, karena Tugas Akhir ini berbasis Kebutuhan Teknologi Informasi. Selanjutnya pada proses *System Design*, Spesifikasi Kebutuhan dari tahap sebelumnya akan dipelajari untuk menyiapkan Desain Sistem dimana Desain Sistem membantu dalam mendefinisikan arsitektur sistem secara keseluruhan. Kemudian pada bagian *Implementation* dilakukanlah penulisan kode program secara nyata untuk menerjemahkan desain yang ada ke dalam bahasa yang bisa di kenali oleh komputer. Setelah semuanya selesai maka akan dilakukan *Testing* terhadap sistem yang telah dibuat untuk menemukan kesalahan-kesalahan pada aplikasi untuk dapat segera diperbaiki dan disempurnakan. Dan ketika aplikasi telah dinyatakan telah siap dan sempurna, maka dilakukanlah tahapan akhir pada metode ini yaitu *Deployment* dimana aplikasi akan disebarkan atau dapat langsung digunakan oleh para penggunanya [26]. Dimana pada Tugas Akhir ini pengguna utamanya yaitu diantaranya *Developer* situs *eCommerce* (Direktori Aplikasi / *Software Directory*) agar dapat melakukan Pengembangan Sistem dan Konten dan juga mengelola Struktur Data Produk dengan mudah. Dan juga agar Pemilik situs *eCommerce* (Direktori Aplikasi / *Software Directory*) melakukan Proses Perubahan dan Pengelolaan Struktur Data Produk dengan mudah dan cepat. Dan yang terakhir yaitu Perusahaan *Search Engine* agar Data dan Informasi Produk Perangkat Lunak dapat dibaca dengan mudah oleh *Search Engine* untuk menghasilkan pencarian yang relevan bagi para penggunananya.

3.1.4. Pengujian Aplikasi

Pada tahap ini dilakukan setelah aplikasi yang dikembangkan selesai. Aplikasi akan dilakukan *Compability Test* yaitu aplikasi akan dijalaan pada skenario pengujian atau *test case* pada beberapa *web browser* populer saat ini diantaranya seperti Safari, Mozilla Firefox dan Google Chrome. Selain itu juga dilakukan *Black Box Testing* untuk memastikan bahwa semua

fungsi dan interaksi yang ada pada aplikasi telah berjalan dengan baik dan sesuai.

3.2. Penyusunan Laporan

Tahap ini merupakan tahapan terakhir dalam pengerjaan tugas akhir. Tahapan ini mendokumentasikan seluruh tahapan yang dilakukan dan seluruh luaran dari setiap proses yang dijalani. Luaran dari proses ini adalah buku Laporan tugas akhir yang disesuaikan dengan format yang sudah ditetapkan oleh Departemen Sistem Informasi.

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN

Pada bab ini membahas alur perancangan terkait beberapa hal yang diperlukan dalam proses pembuatan aplikasi sesuai dengan alur yang dijelaskan pada bab 3. Dalam bab perancangan ini akan menjelaskan tentang proses penggalian kebutuhan dan desain sistem.

4.1. Analisis Kebutuhan Sistem

4.1.1. Analisis Kebutuhan non-Fungsional

Pada tahapan ini dianalisis kebutuhan non-Fungsional dari aplikasi. Kebutuhan non-Functional merupakan kebutuhan yang tidak terkait fungsi dari sistem. Kebutuhan ini mencakup hal – hal seperti kompabilitas tampilan dan keandalan dari website. Tabel 4.1 Analisis Kebutuhan non-Fungsional merupakan hasil analisis kebutuhan non-fungsional dari aplikasi.

Tabel 4.1 Analisis Kebutuhan non-Fungsional

Cross-browser Compability	Aplikasi yang dikembangkan dapat berjalan dan bekerja pada *web broser* yang berbeda-beda diantaranya seperti Safari, Mozilla Firefox dan Google Chrome agar setiap fungsi dan interaksi yang dilakukan oleh pengguna dapat di proses dengan baik oleh aplikasi.
MVC (Model-View-Controller)	Aplikasi yang dikembangkan menggunakan konsep MVC dengan memisahkan aplikasi dalam tiga komponen utama Logis: *Model*, *View* dan *Controller* untuk mempermudah

	pengembangan aplikasi di masa depan.

4.1.2. Analisis Rancangan Fitur

Pada tahapan ini, dilakukan analisis terhadap fitur utama apa saja yang akan dimasukkan ke dalam aplikasi yang akan dibuat. Analisis ini ditentukan berdasarkan pada tahapan studi literatur dengan membandingkan dan mengolaborasikan beberapa aplikasi yang menjadi rujukan dari penelitian. Secara keseluruhan Fitur utama yang diharapkan ada pada aplikasi berserta referensi dari fitur yang ada dijabarkan pada Tabel 4.2 Analisis Rancangan Fitur.

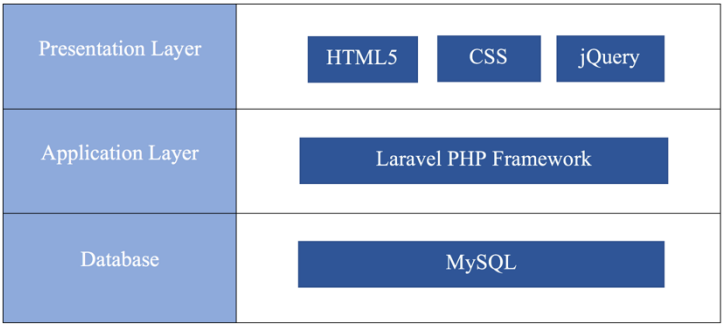
Tabel 4.2 Analisis Rancangan Fitur

No	Fitur	Deskripsi
1	*Add Schema*	Memungkinkan Admin untuk menambahkan Struktur Data atau Struktur Katalog Produk Perangkat Lunak.
2	*Schema Element*	Memungkinkan Admin untuk Menambahkan, Mengubah maupun Menghapus Element pada Struktur Data atau Struktur Katalog Produk Perangkat Lunak yang ada seperti *Text, Image, List, Link* dan *Inlink*.
3	*Schema Versioning*	Memungkinkan Admin untuk melakukan *Restore Schema* atau mengembalikan Struktur Data ke versi-versi sebelumnya baik sebelum maupun sesudah perubahan dilakukan.

No	Fitur	Deskripsi
4	*List Schema*	Menampilkan seluruh Skema yang pernah di buat berdasarkan Status Penerbitannya seperti *Published, Draft* dan *Removed*.
5	*Add Content*	Memungkinkan Pengguna untuk Menambahkan Konten atau Produk Perangkat Lunak Baru.
6	*Content Element*	Memungkinkan Pengguna untuk Mengubah Data / Informasi pada Struktur Data atau Struktur Katalog Produk Perangkat Lunak yang ada pada Konten seperti *Text, Image, List, Link* dan *Inlink*.
7	*Content Versioning*	Memungkinkan Pengguna untuk melakukan *Restore Version* terkait Konten maupun Informasi yang diterbitkan untuk mengembalikan Konten maupun Informasi yang ada ke versi-versi sebelumnya baik sebelum maupun sesudah perubahan dilakukan.
8	*Content Schema*	Memungkinkan Pengguna untuk Menambahkan dan Menghapus Element pada Struktur Data atau Struktur Katalog Produk Perangkat Lunak yang ada seperti *Text, Image, List, Link* dan *Inlink* yang terdapat pada sebuah Konten atau Produk Perangkat Lunak.
9	*Content Collaboration*	Memungkinkan Pengguna untuk saling bergabung dan berkolaborasi untuk meningkatkan kualitas Konten

No	Fitur	Deskripsi
		atau Informasi yang ada pada sebuah Konten atau Produk Perangkat Lunak.
10	*User Block*	Memungkinkan Pengguna untuk melakukan Pemblokiran pada Pengguna dalam sebuah Konten atau Kolaborasi untuk menghindarkan perubahan atau perusakan Konten atau Informasi yang ada dari Pengguna tersebut.
11	*List Content*	Menampilkan seluruh Konten yang pernah di buat berdasarkan Status Penerbitannya seperti *Published, Draft* dan *Removed*.
12	*Search*	Memungkinkan Pengunjung / Pengguna untuk Melakukan Pencarian terhadap Konten yang ada dan juga Admin untuk dapat Melakukan Pencarian terhadap Skema yang ada.
13	*Register*	Memungkinkan Pengunjung untuk mendaftar atau bergabung sebagai *Member* dari aplikasi.
14	*Login*	Memungkinkan Pengguna untuk masuk ke aplikasi.
15	*Profile*	Memungkinkan Pengunjung / Pengguna untuk mengetahui Konten apa saja yang telah diterbitkan oleh seorang pengguna maupun Konten apa saja yang dia miliki.

4.2. Desain Sistem



Gambar 4.1 Arsitektur Teknologi

Gambar 4.1 Arsitektur Teknologi pada aplikasi mengacu pada kerangka kerja yang ada pada Aplikasi Web. Dimana fondasi utama aplikasi ini dibangun menggunakan Laravel 5.4, yang merupakan sebuah *PHP Framework* yang dibangun di atas lebih dari 20 *library* yang berbeda dan dibagi menjadi modul-modul secara individu dan terintegrasi dengan *Composer dependency manager* [24].

Dengan banyaknya fitur-fitur tersebut, produktifitas pengembangan perangkat lunak dapat jauh meningkat. Sehingga di masa depan apabila terjadi penambahan atau pembaruan, pengembangan dapat dilakukan dengan jauh lebih mudah dan terintegrasi.

Basis data yang digunakan pada aplikasi ini yaitu menggunakan MySQL yang merupakan sebuah *Relational Database Management System (RDBMS)* dengan kehandalan dan performanya yang tinggi dimana pengembangannya kian pesat menghadirkan berbagai fitur dan teknologi baru hingga saat ini. MySQL memungkinkan untuk secara efisien menyimpan, mengurutkan dan menampilkan data dari basi data [24]. MySQL mendukung berbagai fitur seperti *multithreaded*, *multi-user*, dan *SQL database managemen sistem (DBMS)*. Basis data

ini dibuat untuk keperluan sistem basis data yang cepat, handal dan mudah digunakan.

Penerapan *Relational Database Management System (RDBMS)* di sini dapat memungkinkan Data dan Informasi pada Aplikasi yang meskipun memiliki Struktur Data yang berbeda-beda namun tetap dapat terhubung dan berelasi sehingga konsistensi dan ketepatan nya tetap terjaga.

4.3. Desain Basis Data

Dalam penyusunan aplikasi pada Tugas Akhir ini digunakan Skema Basis Data sebagai berikut :

Gambar 4.2 Desain Basis Data

Pada Skema Basis Data yang digunakan dalam pengembangan aplikasi pada penelitian ini terdapat 12 Entitas di dalamnya dimana seperti terlihat pada Gambar 4.2 Desain Basis Data.

Pada Skema tersebut terdapat Skema Utama yaitu *"o_schemas"* yang saling beselasi dengan Entitas lainnya yang terkait dengan mengelolaan Struktur Data atau Struktur Katalog Produk Perangkat Lunak pada aplikasi seperti *"o_schema_elements"* yang menyimpan informasi terkait Element apa saja yang dapat digunakan, dan juga terdapat *"o_schema_versions"* yang

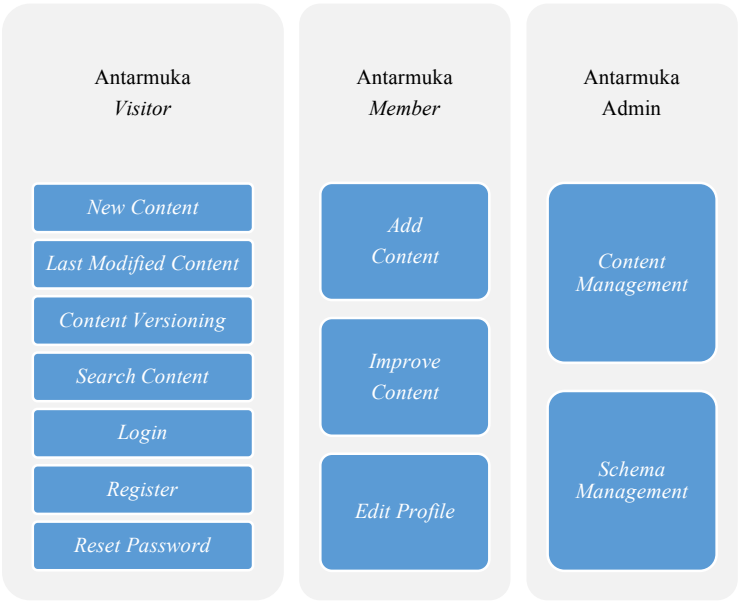
merelasikan versi berapa saja yang tersedia pada sebuah Skema dan versi berapa yang sedang aktif dan yang terakhir yaitu *"o_schema_relations"* yang memegang peran penting dalam menghubungkan Skema dengan Versi Skema dan Element yang ada pada sebuah Skema.

Dimana setelah semua Entitas pada Skema tersebut terhubung dan saling berelasi, nantinya Skema tersebutlah yang menjadi referensi dari Entitas Konten diantaranya *"o_contents"* , *"o_content_relations"* dan *"o_content_versions"* untuk melakukan *cloning* dari Entitas Skema yang menjadi referensi ke Entitas Konten tersebut. Yang menjadi pembeda adalah pada Entitas *"o_content_relations"* terdapat relasi terhadap *"o_content_values"* yang mengatur data apa saja yang tersedia dalam sebuah Element terkait tersedianya versi berapa saja dan versi mana yang sedang aktif, hal tersebut berfungsi untuk mengakomodir fitur *Content Versioning* yang terkait dengan *Restore Version*. Selain itu terdapat juga Entitas *"o_content_blocks"* yang berfungsi untuk menyimpan informasi pengguna siapa saja yang tidak diberikan akses ke dalam sebuah konten tertentu untuk melakukan perubahan di dalamnya.

Setiap Entitas yang ada tersebut akan berelasi dengan Entitas *"users"* agar setiap aktifitas dan perubahan Entitas yang ada dapat terekam dengan pasti kapan terjadinya dan siapa yang melakukannya. Dan setiap Entitas yang berkaitan dengan Skema dan Konten akan menggunakan *prefix* atau awalan *"o_"* untuk memudahkan pengembangan di masa depan untuk mendefinisikan fungsi dari setiap Entitas yang ada dalam Struktur Basis Data.

4.4. Desain Antarmuka

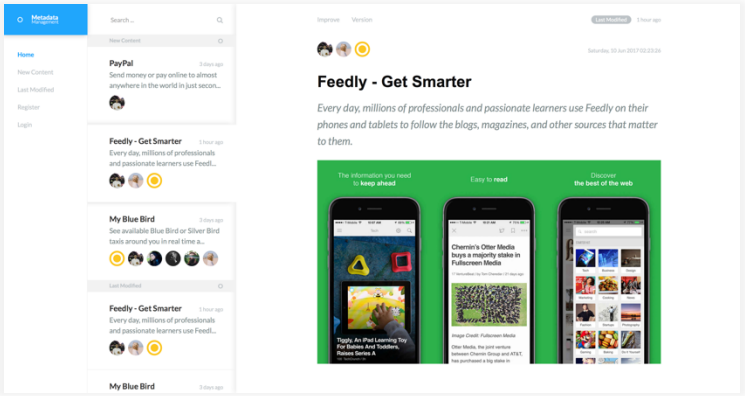
Antarmuka pada aplikasi akan dibagi menjadi 3 Bagian berdasarkan Peran Pengguna pada aplikasi diantaranya Antarmuka *Visitor*, Antarmuka *Member* dan Antarmuka Admin yang dapat dilihat pada Gambar 4.3 Desain Antarmuka.



Gambar 4.3 Desain Antarmuka

4.4.1. Antarmuka *Visitor*

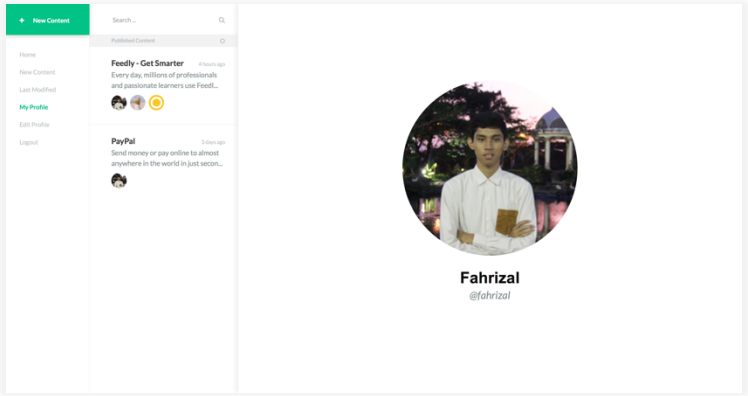
Visitor merupakan pengunjung aplikasi secara umum yang belum menjadi *Member* atau *Member* yang belum *Login* ke aplikasi. Dimana pada Antarmuka *Visitor*, *Visitor* atau Pengunjung secara umum memiliki akses terhadap beberapa Halaman diantaranya Halaman Utama Aplikasi, *New Content, Last Modified Content, Content Versioning, Search Content, Login, Register* dan *Reset Password* seperti tampak pada Gambar 4.4 Antarmuka Visitor.



Gambar 4.4 Antarmuka Visitor

4.4.2. Antarmuka *Member*

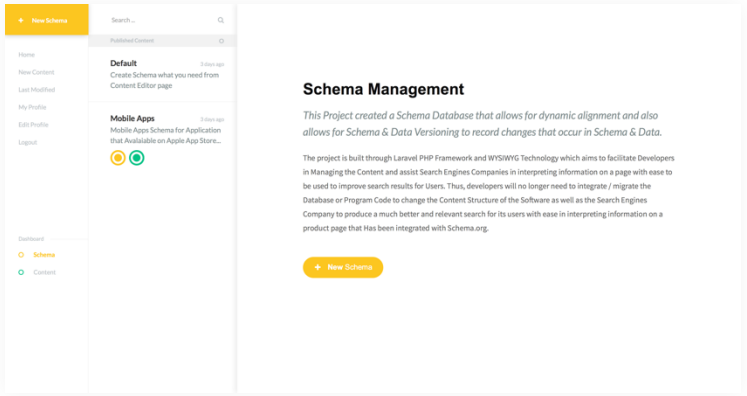
Member merupakan pengunjung yang telah *Register* atau terdaftar pada aplikasi dimana setiap pengunjung yang telah menjadi Member akan mendapatkan keuntungan dan memiliki akses pada Antarmuka *Member* diantaranya dapat Menambahkan Konten atau Produk Perangkat Lunak Baru (*New Content*), Mengembangkan atau Berkolaborasi meningkatkan kualitas Produk Perangkat Lunak yang telah diterbitkan (*Edit Content*) dan juga tentunya Mengubah / Memperbaruhi Informasi Profil Member (*Edit Profile*) seperti tampak pada Gambar 4.5 Antarmuka Member.



Gambar 4.5 Antarmuka Member

4.4.3. Antarmuka Admin

Admin merupakan Pengelola Aplikasi atau dapat dikatakan sebagai Pemilik Aplikasi / Situs dimana Admin memiliki kontrok penuh terhadap aplikasi dimana Admin memilik peran penuh baik sebagai Pengunjung, *Member* dan juga Pemilik Situs. Admin memiliki akses penuh untuk dapat Mengelola seluruh Konten atau Produk Perangkat Lunak Baru (*Content Management*) dan Mengelola seluruh Skema Produk Perangkat Lunak Baru ataupun Konten lainnya (*Schema Management*) seperti tampak pada Gambar 4.6 Antarmuka Admin.



Gambar 4.6 Antarmuka Admin

4.5. Variasi Data

Dalam Pengelolaan Katalog Produk Perangkat Lunak pada Aplikasi, terdapat Variasi Data yang digunakan dalam Kerangka Produk Perangkat Lunak yang mengacu berdasarkan Variasi Data pada *Schema.org* [4]. Seperti tampak pada Tabel 4.3 Variasi Data terdapat 28 Variasi Data yang digunakan dalam sebuah Struktur Data Perangkat Lunak berdasarkan *Schema.org*.

Tabel 4.3 Variasi Data

No	Tipe Data	Deskripsi
1	*Text*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Tulisan.
2	*External URL*	Jenis Data ini memungkinkan untuk menambahkan sebuah

No	Tipe Data	Deskripsi
		Struktur Data dalam bentuk URL ke luar aplikasi.
3	*SoftwareApplication / Internal URL*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk URL yang berelasi dengan URL Produk Perangkat Lunak lainnya pada Aplikasi.
4	*Image*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Gambar.
5	*DataFeed / List*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Daftar.
6	*Person*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan Informasi atau Profil Seseorang.
7	*AggregateRating*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Rating atau Ulasan pada Produk Perangkat Lunak.
8	*Video*	Jenis Data ini memungkinkan untuk menambahkan sebuah

No	Tipe Data	Deskripsi
		Struktur Data dalam bentuk *Video*.
9	*Audience*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan target Pengguna dari Produk Perangkat Lunak.
10	*Audio*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk *Audio*.
11	*Organization*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan Informasi atau Profil sebuah Perusahaan.
12	*CreativeWork*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan *CreativeWork* yang terkait pada Produk Perangkat Lunak.
13	*Comment*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Komentar dari Pengguna.

No	Tipe Data	Deskripsi
14	*Integer*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk *Integer*.
15	*Place*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan informasi suatu Lokasi.
16	*Number*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Nomor.
17	*Date*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Tanggal.
18	*DateTime*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Tanggal dan Waktu.
19	*AlignmentObject*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan Sumber Informasi.
20	*InteractionCounter*	Jenis Data ini memungkinkan untuk menambahkan sebuah

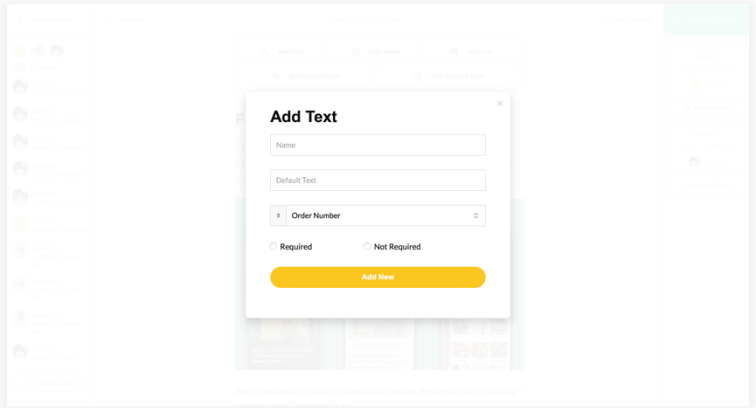
No	Tipe Data	Deskripsi
		Struktur Data yang berhubungan dengan Penghitungan dari Interaksi yang dilakukan oleh Pengguna.
21	*Language*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan Bahasa.
22	*Boolean*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk *Boolean.*
23	*PublicationEvent*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Publikasi dari suatu Kegiatan.
24	*Event*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data yang berhubungan dengan Informasi terkait suatu Kegiatan.
25	*Review*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Ulasa.
26	*VideoObject*	Jenis Data ini memungkinkan untuk menambahkan sebuah

No	Tipe Data	Deskripsi
		Struktur Data dalam bentuk Video.
27	*PropertyValue*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk informasi yang bersifat tunggal pada Produk Perangkat Lunak yang nantinya akan terbaca dalam sebuah Deskripsi dari Produk Perangkat Lunak.
28	*Action*	Jenis Data ini memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Tindakan yang dimungkinkan oleh pengguna.

Dari 28 Variasai Data yang digunakan oleh *Schema.org* dan sudah dijelaskan pada Tabel 4.3 Variasi Data akan digunakan 5 Tipe Data sebagai kerangka dasar dari Pengelolaan Struktur Data pada Tugas Akhir ini diantaranya: *Text, Enternal URL, Internal URL, Image* dan *List.*

4.5.1. Tipe Data *Text*

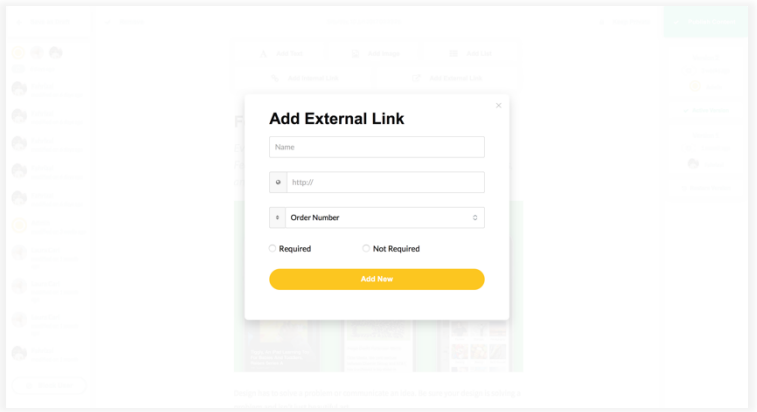
Text merupakan Jenis Data yang memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Tulisan seperti tampak pada Gambar 4.7 Tipe Data Text.



Gambar 4.7 Tipe Data Text

4.5.2. Tipe Data *External URL*

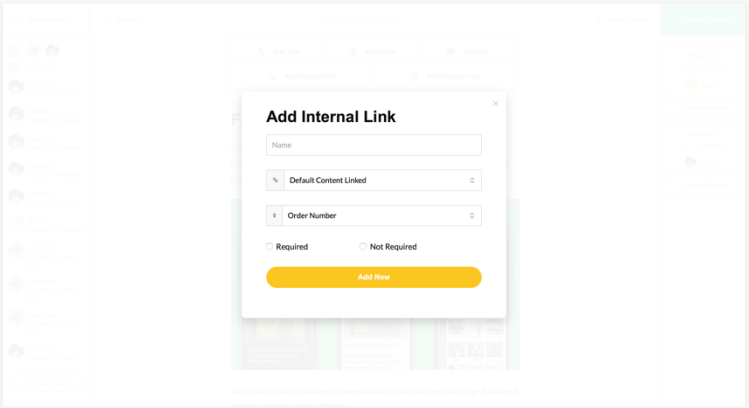
External URL merupakan Jenis Data yang memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk URL ke luar aplikasi seperti tampak pada Gambar 4.8 Tipe Data External URL.



Gambar 4.8 Tipe Data External URL

4.5.3. Tipe Data *Internal URL*

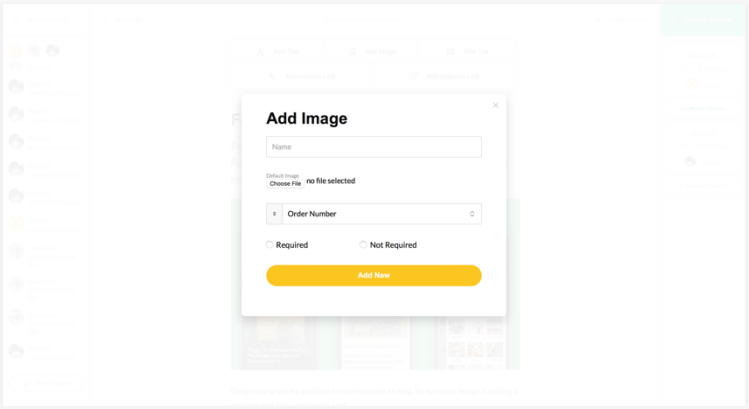
Internal URL merupakan Jenis Data yang memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk URL yang berelasi dengan URL Produk Perangkat Lunak lainnya pada Aplikasi seperti tampak pada Gambar 4.9 Tipe Data Internal URL.



Gambar 4.9 Tipe Data Internal URL

4.5.4. Tipe Data *Image*

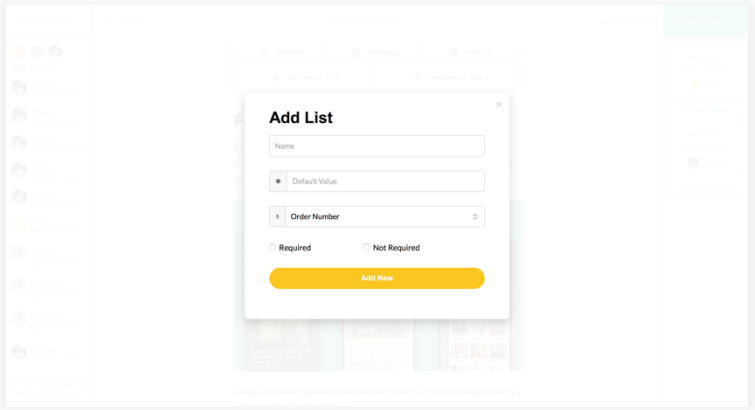
Image merupakan Jenis Data yang memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk Gambar seperti tampak pada Gambar 4.10 Tipe Data Image.



Gambar 4.10 Tipe Data Image

4.5.5. Tipe Data *List*

Image merupakan Jenis Data yang memungkinkan untuk menambahkan sebuah Struktur Data dalam bentuk sebuah Daftar seperti tampak pada Gambar 4.11 Tipe Data List.



Gambar 4.11 Tipe Data List

BAB V
IMPLEMENTASI

Pada bab ini akan dijelaskan terkait proses implementasi pada perangkat lunak yang telah dirancang pada bab sebelumnya.

5.1. Lingkungan Implementasi

Pada pengembangan aplikasi pada penelitian ini, memanfaatkan perangkat keras dengan spesifikasi seperti pada Tabel 5.1 Spesifikasi Perangkat Keras. Sedangkan untuk perangkat lunak yang digunakan dalam pengembangan aplikasi adalah seperti pada Tabel 5.2 Spesifikasi Perangkat Lunak.

Tabel 5.1 Spesifikasi Perangkat Keras

Perangkat Keras	**Spesifikasi**
Sistem Operasi	macOS Sierra
Processor	2.2 GHz Intel Core i7
RAM	16 GB 1600 MHz DDR3
Flash Storage	256 GB

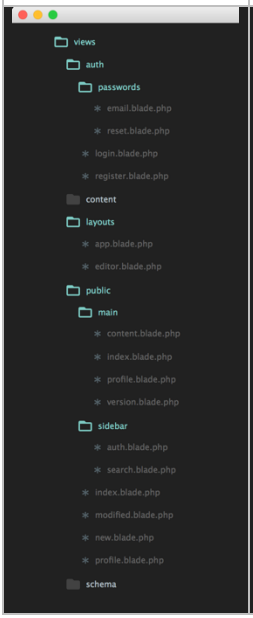
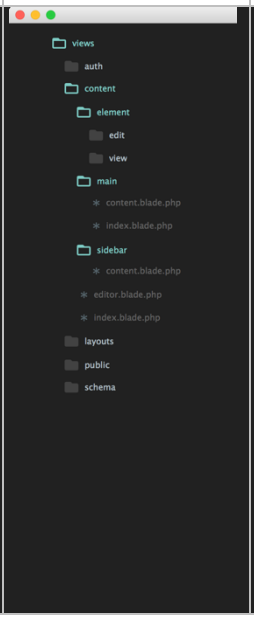
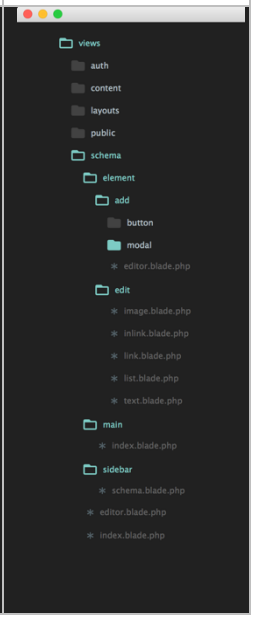
Tabel 5.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	**Spesifikasi**
Apache Version	2.4.12
MySQL Version	5.6.25
PHP Version	5.6.4
PHP Framework	Laravel 5.4
PHP Extension	OpenSSL, PDO PHP, Mbstring, Tokenizer, XML PHP, Fileinfo, GD Library
Text Editor	Sublime Text 3126

Perangkat Lunak	Spesifikasi
Web Browser	Safari 10.1.1, Firefox Developer Edition 55, Google Chrome 59

5.2. Struktur Direktori

Pada pengembangan aplikasi pada penelitian ini memanfaatkan *Framework* Laravel 5.4, sehinggi Struktur Direktori Utama aplikasi akan mengikuti Struktur Direktori *Framework* yang digunakan tersebut. Namun, terdapat sebuah direktori yang memiliki peran penting dan bergantung terhadap kualitas pengembang dalam mengelola struktur direktori pada sebuah aplikasi yaitu Direktori *Views* karena direktori tersebut memegang peran penting dalam pengembangan aplikasi di masa depan terutama apabila berkaitan lebih dalam pada Antarmuka Aplikasi. Oleh karena itu, Struktur Direktori *Views* dibagi menjadi 3 bagian diantaranya yang bersifat *Public* atau lebih umum yang berkaitan dengan Halaman Pengguna secara Umum maupun yang bersifat autentikasi pada aplikasi, selanjutnya *Content* yang berkaitan dengan *Content Management* pada Aplikasi, begitu juga dengan *Schema* yang berkaitan dengan *Schema Management* seperti tampak pada Gambar 5.1 Struktur Direktori.

Direktori *Public*	Direktori *Content*	Direktori *Schema*
		

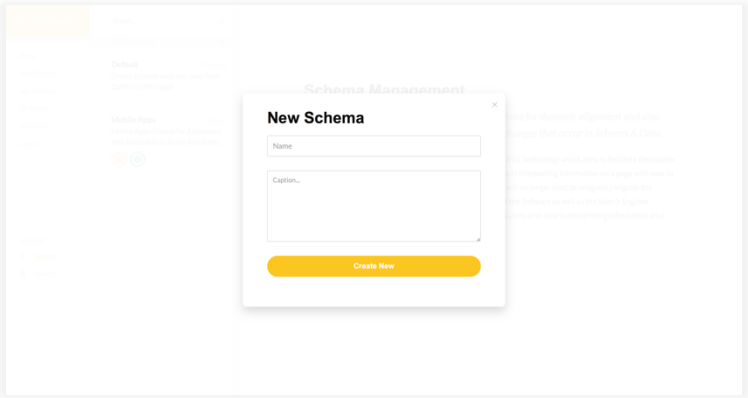
Gambar 5.1 Struktur Direktori

5.3. Implementasi Aplikasi

5.3.1. Implementasi Fitur *Add Schema*

Fitur *Add Schema* merupakan sebuah fitur utama yang menjadi kunci dari semua fitur yang ada, karena hasil luaran dari fitur inilah yang menjadi penghubung dari setiap Struktur Data pada aplikasi dan menjadi referensi dari setiap Konten yang ada. Informasi yang dibutuhkan untuk membuat sebuah skema baru yaitu cukup mengisi Nama dan Keterangan dari Skema Baru

tersebut seperti tampak pada Gambar 5.2 Implementasi Fitur Add Schema.



Gambar 5.2 Implementasi Fitur Add Schema

Selanjutnya, sistem akan melakukan validasi terhadap masukan yang ada untuk kemudian dibuatlah Skema dan sebuah informasi terkait Versi dari Skema Baru tersebut untuk mendukung Fitur *Schema Versioning* seperti tampak pada Gambar 5.3 Kode Program Add Schema. Pada *function validator* dilakukan validasi masukan dari pengguna yang berupa *name* dan *caption*. Selanjutnya masukan tersebut akan di proses seperti tampak pada *function create* dari dimulainya *transaction* pada bari ke 153 sampai 174 dimana akan disiapkan sebuah informasi terkait *schema* baru dan *schema version* baru yang saling berelasi.

File: app/Http/Controllers/SchemaController.php

```
132:     protected function validator(array $data)
133:     {
134:                 return Validator::make($data, [
135:                             'name' => 'required|max:255',
136:                             'caption' => 'required',
137:                 ]);
138:     }
139:
140:    public function create(Request $request)
141:    {
142:                 $validator    =    $this->validator($request-
>all());
143:
144:         if ($validator->fails())
145:         {
146:             $this->throwValidationException(
147:                 $request, $validator
148:             );
149:         }
150:
151:                 $data = $request->all();
152:
153:                 DB::beginTransaction();
154:
```

```
155:            $schemaVersion = new SchemaVersion;
156:            $schemaVersion->user_id = Auth::user()->id;
157:            $schemaVersion->number = 1;
158:            $schemaVersion->save();
159:
160:            $schema = new Schema;
161:            $schema->user_id = Auth::user()->id;
162:            $schema->name = $data['name'];
163:            $schema->slug = str_slug($data['name']);
164:            $schema->caption = $data['caption'];
165:            $schema->active = 1;
166:            $schema->version_id = $schemaVersion->id;
167:            $schema->save();
168:
169:            $schemaId = $schema->id;
170:
171:                $schemaVersionUpdate = SchemaVersion::where('id', $schemaVersion->id)
172:        ->update(['schema_id' => $schemaId]);
173:
174:            DB::commit();
175:
176:            return redirect('schema/'.$schemaId);
```

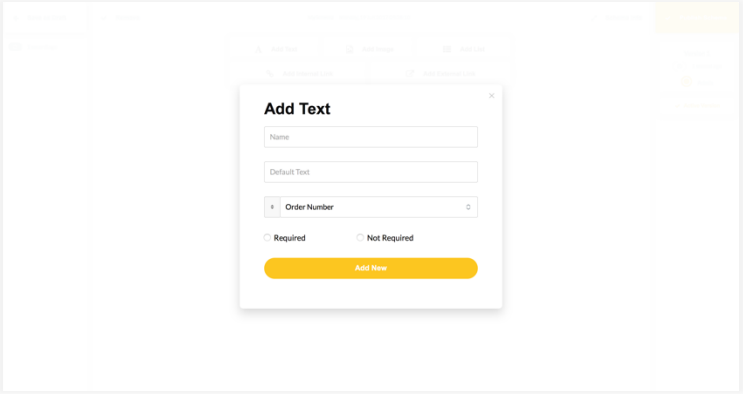
```
177:    }
```

Gambar 5.3 Kode Program Add Schema

5.3.2. Implementasi Fitur *Schema Element*

5.3.2.1. Implementasi Fitur *Add Schema Element*

Fitur *Add Schema Element* merupakan sebuah fitur yang memungkinkan untuk melakukan penambahan *Element* pada sebuah *Schema* atau Struktur Data seperti tampak pada Gambar 5.4 Implementasi Fitur Add Schema Element dimana untuk menambahkan sebuah *Add Schema* ada beberapa informasi yang harus di lengkapi terkait Nama *Element*, *Default Value* dari *Element* tersebut, urutan *Element* tersebut dalam sebuah *Schema* atau Struktur Data dan juga informasi terkait apakah *Element* tersebut harus di isi atau bersifat *Optional* apabila nantinya ada sebuah Konten yang akan menggunakannya.



Gambar 5.4 Implementasi Fitur Add Schema Element

Kemudian, sistem akan melakukan validasi dari masukan yang ada tersebut untuk membuat sebuah Versi Baru dari Skema dan membuah sebuah Relasi Baru untuk menghubungkan Skema dan Versi barunya dengan *Element* yang akan di tambahkan tersebut. Dan yang terpenting yaitu akan dilakukan proses

Schema Cloning dari Relasi Skema yang lama ke Versi yang baru seperti tampak pada Gambar 5.5 Kode Program Add Schema Element, hal tersebut perlu dilakukan untuk mendukung Fitur *Schema Versioning*.

File: app/Http/Controllers/SchemaController.php

```
249:    protected function validatorAdd(array $data)
250:    {
251:        return Validator::make($data, [
252:            'schema_id' => 'required|integer',
253:            'element_id' => 'required|integer',
254:            'active' => 'required|integer',
255:            'name' => 'required|max:255',
256:            'default_value' => 'required',
257:            'order_number' => 'required|integer',
258:            'required' => 'required|integer',
259:        ]);
260:    }
261:
262:    public function add(Request $request)
263:    {
264:        $validator = $this->validatorAdd($request->all());
265:
266:        if ($validator->fails())
267:        {
268:            $this->throwValidationException(
```

```
269:                $request, $validator
270:            );
271:        }
272:
273:        $data = $request->all();
274:
275:        DB::beginTransaction();
276:
277:        $schemaId = $data['schema_id'];
278:        $elementId = $data['element_id'];
279:
280:            $defaultValue = $data['default_value']; // Text
Value
281:        if ($elementId == 2) // Image Value
282:        {
283:            $validator = Validator::make($request->all(), [
284:                                                'default_value'   =>
'required|image|mimes:jpeg,png,jpg,gif|max:2048'
285:            ]);
286:
287:            if ($validator->fails())
288:            {
289:                return redirect('schema/'.$schemaId)->with([
290:                    'warning' => 'Terjadi Kesalahan!! Ukuran
Foto Maksimal : 2MB',
```

```
291:                ]);
292:            }
293:
294:                        $imageName = md5(Auth::user()->id.Auth::user()->username.$request->default_value->getClientOriginalName()).'.'.$request->default_value->getClientOriginalExtension();
295:                                $request->default_value->move(public_path('uploads/image/'), $imageName);
296:
297:            $defaultValue = $imageName;
298:        }
299:
300:        $schemaCheck = Schema::where('id', $schemaId)->first();
301:
302:                                $schemaVersionCount = SchemaVersion::where('schema_id', $schemaId)
303:            ->count();
304:
305:        $schemaVersion = new SchemaVersion;
306:        $schemaVersion->user_id = Auth::user()->id;
307:                                $schemaVersion->number = $schemaVersionCount+1;
308:        $schemaVersion->schema_id = $schemaId;
309:        $schemaVersion->save();
310:
```

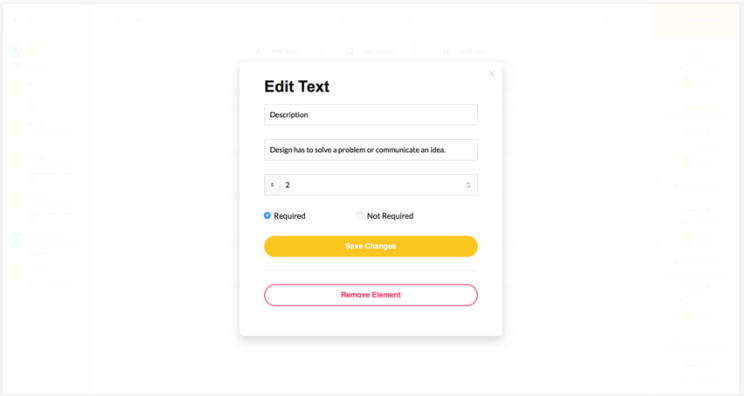
```
311:        $versionIdOld = $schemaCheck->version_id;
312:        $versionId = $schemaVersion->id;
313:
314:        $schemaUpdate = Schema::where('id', $schemaId)
315:            ->update(['version_id' => $versionId]);
316:
317:                                    $schemaRelationCopy =
SchemaRelation::where('schema_id', $schemaId)
318:            ->where('version_id', $versionIdOld)
319:            ->get();
320:
321:        foreach ($schemaRelationCopy as $relation)
322:        {
323:            $schemaRelation = new SchemaRelation;
324:           $schemaRelation->user_id = $relation->user_id;
325:            $schemaRelation->name = $relation->name;
326:              $schemaRelation->default_value = $relation-
>default_value;
327:                   $schemaRelation->required = $relation-
>required;
328:            $schemaRelation->active = $relation->active;
329:             $schemaRelation->order_number = $relation-
>order_number;
330:                $schemaRelation->element_id = $relation-
>element_id;;
```

```
331:                $schemaRelation->schema_id = $relation->schema_id;
332:            $schemaRelation->version_id = $versionId;
333:            $schemaRelation->save();
334:        }
335:
336:        $schemaRelation = new SchemaRelation;
337:        $schemaRelation->user_id = Auth::user()->id;
338:        $schemaRelation->name = $data['name'];
339:        $schemaRelation->default_value = $defaultValue;
340:        $schemaRelation->required = $data['required'];
341:        $schemaRelation->active = $data['active'];
342:                        $schemaRelation->order_number = $data['order_number'];
343:                            $schemaRelation->element_id = $data['element_id'];
344:      $schemaRelation->schema_id = $data['schema_id'];
345:        $schemaRelation->version_id = $versionId;
346:        $schemaRelation->save();
347:
348:        DB::commit();
349:
350:        return redirect('schema/'.$schemaId);
351:    }
```

Gambar 5.5 Kode Program Add Schema Element

5.3.2.2. Implementasi Fitur *Edit Schema Element*

Fitur *Edit Schema Element* merupakan sebuah fitur yang memungkinkan untuk melakukan perubahan atau pembaruhan *Element* pada sebuah *Schema* atau Struktur Data seperti tampak pada Gambar 5.6 Implementasi Fitur Edit Schema Element terkait informasi Nama *Element*, *Default Value* dari *Element* tersebut, urutan *Element* tersebut dalam sebuah *Schema* atau Struktur Data dan juga informasi terkait apakah *Element* tersebut harus di isi atau bersifat *Optional*.



Gambar 5.6 Implementasi Fitur Edit Schema Element

Kemudian, sistem akan melakukan validasi dari perubahan yang ada tersebut untuk memperbaruhi informasi terkait *Schema Relation* dari *Element* tersebut seperti tampak pada Gambar 5.7 Kode Program Edit Schema Element setiap masukan dan informasi terkait *schema* yang akan di ubah akan dilakukan validasi pada *function validatorEdit*. Kemudian selanjutnya perubahan akan di spesifikkan berdasarkan Jenis *Element* atau Tipe Data yang dilakukan perubahan dimana setelah proses pada *function edit* tersebut telah selesai maka perubahan akan di simpan pada *schema relation*.

File: app/Http/Controllers/SchemaController.php

```
353:    protected function validatorEdit(array $data)
354:    {
355:        return Validator::make($data, [
356:            'schema_id' => 'required|integer',
357:            'element_id' => 'required|integer',
358:            'active' => 'required|integer',
359:            'name' => 'required|max:255',
360:            'order_number' => 'required|integer',
361:            'required' => 'required|integer',
362:        ]);
363:    }
364:
365:    public function edit(Request $request)
366:    {
367:                $validator = $this->validatorEdit($request->all());
368:
369:         if ($validator->fails())
370:        {
371:            $this->throwValidationException(
372:                $request, $validator
373:            );
374:         }
375:
```

```
376:                $data = $request->all();
377:
378:        $schemaId = $data['schema_id'];
379:        $elementId = $data['element_id'];
380:        $relationId = $data['relation_id'];
381:
382:                                $schemaRelationCheck = SchemaRelation::where('id', $relationId)
383:            ->first();
384:
385:        if (empty($data['default_value']))
386:        {
387:                $defaultValue = $schemaRelationCheck->default_value; // Text Value
388:        }
389:
390:        if (!empty($data['default_value']))
391:        {
392:            $defaultValue = $data['default_value']; // Text Value
393:        }
394:
395:        if ($elementId == 2) // Image Value
396:        {
397:
```

```
398:            if (!empty($data['default_value']))
399:            {
400:                $validator = Validator::make($request->all(),
[
401:                                                'default_value'   =>
'required|image|mimes:jpeg,png,jpg,gif|max:2048'
402:                ]);
403:
404:                if ($validator->fails())
405:                {
406:                 return redirect('schema/'.$schemaId)->with([
407:                     'warning' => 'Terjadi Kesalahan!! Ukuran
Foto Maksimal : 2MB',
408:                   ]);
409:                }
410:
411:
unlink(public_path('uploads/image/').$schemaRelationChec
k->default_value);
412:
413:                            $imageName = md5(Auth::user()-
>id.Auth::user()->username.$request->default_value-
>getClientOriginalName()).'.'.$request->default_value-
>getClientOriginalExtension();
414:                                        $request->default_value-
>move(public_path('uploads/image/'), $imageName);
415:
```

```
416:                $defaultValue = $imageName;
417:            }
418:
419:        }
420:
421:                                    $schemaRelation     =
SchemaRelation::find($relationId);
422:        $schemaRelation->name = $data['name'];
423:        $schemaRelation->default_value = $defaultValue;
424:        $schemaRelation->required = $data['required'];
425:        $schemaRelation->active = $data['active'];
426:                        $schemaRelation->order_number    =
$data['order_number'];
427:        $schemaRelation->save();
428:
429:        return redirect('schema/'.$schemaId);
430:    }
```

Gambar 5.7 Kode Program Edit Schema Element

5.3.2.3. Implementasi Fitur *Remove Schema Element*

Fitur *Remove Schema Element* merupakan sebuah fitur yang berfungsi untuk menghapus *Element* pada sebuah *Schema* atau Struktur Data. Seperti tampak pada Gambar 5.8 Kode Program Remove Schema Element akan dibuat sebuah Versi Baru dari *Schema* kemudian dilakukan *Schema Cloning* dari Relasi Skema yang lama ke Versi yang baru kecuali Element yang di hapus, hal tersebut perlu dilakukan untuk mendukung Fitur *Schema Versioning*.

File: app/Http/Controllers/SchemaController.php

```
432:    public function rem(Request $request)
433:    {
434:        DB::beginTransaction();
435:
436:        $schemaId = $request->schema_id;
437:        $relationId = $request->relation_id;
438:
439:        $schemaCheck = Schema::where('id', $schemaId)
->first();
440:
441:                                    $schemaVersionCount =
SchemaVersion::where('schema_id', $schemaId)
442:            ->count();
443:
444:        $schemaVersion = new SchemaVersion;
445:        $schemaVersion->user_id = Auth::user()->id;
446:                                $schemaVersion->number =
$schemaVersionCount+1;
447:        $schemaVersion->schema_id = $schemaId;
448:        $schemaVersion->save();
449:
450:        $versionIdOld = $schemaCheck->version_id;
451:        $versionId = $schemaVersion->id;
452:
```

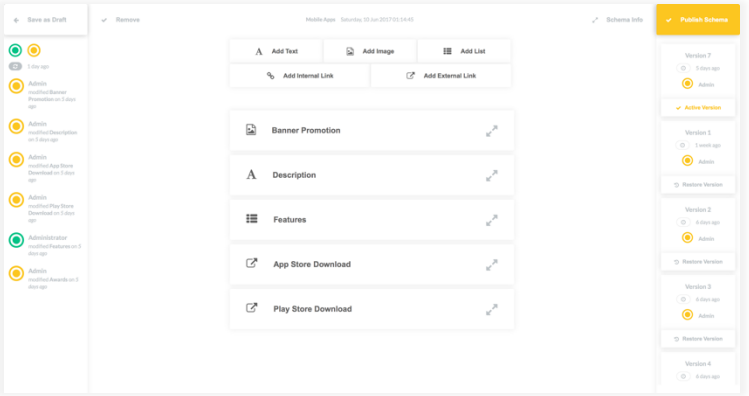
```
453:        $schemaUpdate = Schema::where('id', $schemaId)
454:            ->update(['version_id' => $versionId]);
455:
456:                                    $schemaRelationCopy =
SchemaRelation::where('schema_id', $schemaId)
457:            ->where('version_id', $versionIdOld)
458:            ->get();
459:
460:        foreach ($schemaRelationCopy as $relation)
461:        {
462:            if($relation->id != $relationId)
463:            {
464:                $schemaRelation = new SchemaRelation;
465:                        $schemaRelation->user_id = $relation-
>user_id;
466:                $schemaRelation->name = $relation->name;
467:                $schemaRelation->default_value = $relation-
>default_value;
468:                    $schemaRelation->required = $relation-
>required;
469:                $schemaRelation->active = $relation->active;
470:               $schemaRelation->order_number = $relation-
>order_number;
471:                  $schemaRelation->element_id = $relation-
>element_id;;
472:                   $schemaRelation->schema_id = $relation-
>schema_id;
```

```
473:                $schemaRelation->version_id = $versionId;
474:                $schemaRelation->save();
475:            }
476:        }
477:
478:        DB::commit();
479:
480:        return redirect('schema/'.$schemaId);
481:    }
```

Gambar 5.8 Kode Program Remove Schema Element

5.3.3. Implementasi Fitur *Schema Versioning*

Fitur *Schema Versioning* memungkinkan untuk melakukan *Restore Schema* atau mengembalikan Struktur Data ke versi-versi sebelumnya baik sebelum maupun sesudah perubahan dilakukan, dimana letak fitur ini tepat di bagian kanan layar yang berbentuk sebuah daftar seperti tampak pada Gambar 5.9 Implementasi Fitur Schema Versioning.



Gambar 5.9 Implementasi Fitur Schema Versioning

Dimana ketika salah satu Versi dari Skema tersebut dipilih dari daftar untuk melakukan *Restore Version* maka sistem akan melakukan pembaruan versi dari Versi Skema yang sedang Aktif ke Versi Skema yang dipilih tersebut seperti tampak pada Gambar 5.10 Kode Program Schema Versioning.

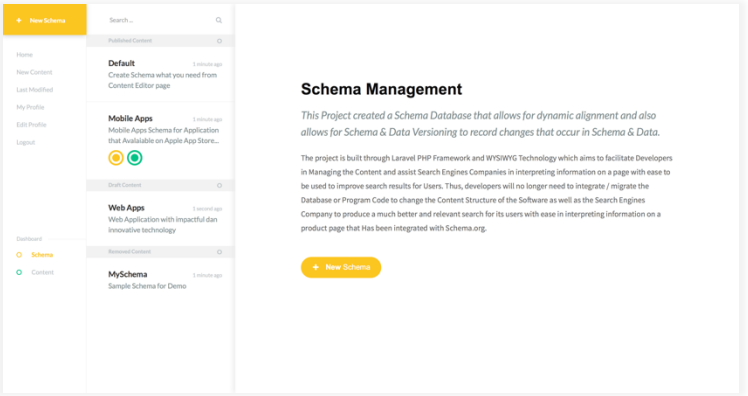
File: app/Http/Controllers/SchemaController.php

```
209:    public function version(Request $request)
210:    {
211:        $schemaId = $request->schema_id;
212:
213:        if (!isset($request->version_id))
214:        {
215:            return redirect('/');
216:        }
217:        $versionId = $request->version_id;
218:
219:        $schema = Schema::where('id', $schemaId)
220:            ->update(['version_id' => $versionId]);
221:
222:        return redirect('schema/'.$schemaId);
223:    }
```

Gambar 5.10 Kode Program Schema Versioning

5.3.4. Implementasi Fitur *List Schema*

Fitur *List Schema* berfungsi untuk menampilkan seluruh Skema yang pernah di buat berdasarkan Status Penerbitannya seperti *Published, Draft* dan *Removed* yang dapat dilihat pada Gambar 5.11 Implementasi Fitur List Schema.



Gambar 5.11 Implementasi Fitur List Schema

Dimana Status Penerbitan dari setiap Skema tersebut dibedakan terdiri dari *active* dan *published* status dengan aturan seperti tampak pada Gambar 5.12 Kode Program List Schema.

File: app/Http/Controllers/SchemaController.php

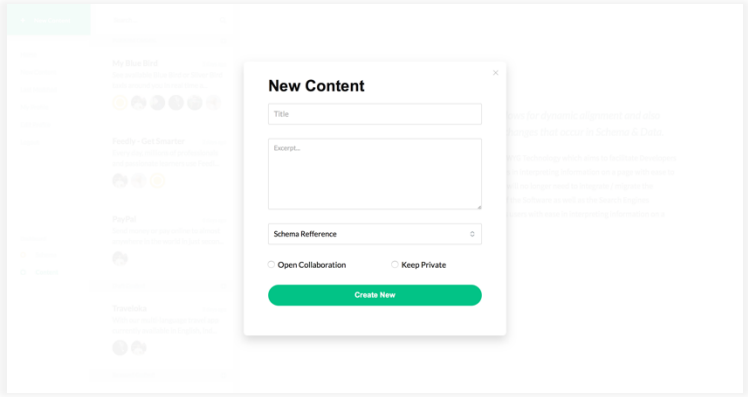
```
18:     public function index()
19:     {
20:         $published = Schema::where('active', '1')
21:             ->where('published', '1')
22:             ->get();
23:
24:         $draft = Schema::where('active', '1')
25:             ->where('published', '0')
26:             ->get();
27:
28:         $removed = Schema::where('active', '0')
29:             ->where('published', '0')
30:             ->get();
31:
32:             $count = count($published) + count($draft) +
count($removed);
33:
34:         return view('schema.index')->with([
35:             'position' => 'comparison',
36:             'published' => $published,
37:             'draft' => $draft,
38:             'removed' => $removed,
39:             'count' => $count,
40:         ]);
```

```
41:     }
```

Gambar 5.12 Kode Program List Schema

5.3.5. Implementasi Fitur *Add Content*

Fitur *Add Content* memungkinkan untuk melakukan penambahan Konten atau Produk Perangkat Lunak Baru, dimana Fitur ini berelasi dengan Fitur Skema yang dimana fitur tersebut akan menjadi referensi dari Konten yang dibuat seperti tampak pada Gambar 5.13 Implementasi Fitur Add Content, hal tersebut bertujuan untuk melakukan *Schema Cloning* dari Skema yang menjadi referensi tersebut ke dalam Konten yang dibuat.



Gambar 5.13 Implementasi Fitur Add Content

Dimana dapat dilihat pada Gambar 5.14 Kode Program Add Content terdapat sebuah proses panjang dimulai dari dibuatnya sebuah Versi Konten dan juga Informasi terkait Konten tersebut dimana Relasi dari Skema pada Konten tersebut akan di inisiasi oleh hasil *Schema Cloning* dari Skema yang menjadi referensi dimana *Default Value* dari Konten akan di definisikan dari *Default Value* yang telah di tentukan pada saat Skema dibuat.

File: app/Http/Controllers/ContentController.php

```
136:    protected function validator(array $data)
137:    {
138:        return Validator::make($data, [
139:            'title' => 'required|max:255',
140:            'excerpt' => 'required',
141:            'schema_id' => 'required|integer',
142:            'customized' => 'required|integer',
143:        ]);
144:    }
145:
146:    public function create(Request $request)
147:    {
148:        $validator = $this->validator($request->all());
149:
150:        if ($validator->fails())
151:        {
152:            $this->throwValidationException(
153:                $request, $validator
154:            );
155:        }
156:
157:        $data = $request->all();
158:
```

```
159:        DB::beginTransaction();
160:
161:        $schemaId = $data['schema_id'];
162:
163:        $contentVersion = new ContentVersion;
164:        $contentVersion->user_id = Auth::user()->id;
165:        $contentVersion->number = 1;
166:        $contentVersion->save();
167:
168:        $content = new Content;
169:        $content->user_id = Auth::user()->id;
170:        $content->title = $data['title'];
171:        $content->slug = str_slug($data['title']);
172:        $content->excerpt = $data['excerpt'];
173:        $content->active = 1;
174:        $content->version_id = $contentVersion->id;
175:        $content->schema_id = $schemaId;
176:        $content->customized = $data['customized'];
177:        $content->published = 0;
178:        $content->save();
179:
180:        $contentId = $content->id;
181:
182:                                $contentVersionUpdate =
ContentVersion::where('id', $contentVersion->id)
```

```
183:            ->update(['content_id' => $contentId]);
184:
185:        $schemaCheck = Schema::where('id', $schemaId)-
>first();
186:
187:                                        $schemaRelations =
SchemaRelation::where('schema_id', $schemaId)
188:                        ->where('version_id',  $schemaCheck-
>version_id)
189:            ->where('active', 1)
190:            ->get();
191:
192:        foreach ($schemaRelations as $relation)
193:        {
194:            $contentValue = new ContentValue;
195:            $contentValue->user_id = Auth::user()->id;
196:            $contentValue->content_id = $contentId;
197:
198:            if ($relation->element_id == 1)
199:            {
200:                                $contentValue->text = $relation-
>default_value;
201:            }
202:
203:            if ($relation->element_id == 2)
204:            {
```

```
205:                    $oldPath = 'uploads/image/'.$relation->default_value;
206:                                                    $newPath = 'uploads/image/'.$contentId.$relation->default_value;
207:            \File::copy($oldPath , $newPath);
208:
209:              $contentValue->file = $contentId.$relation->default_value;
210:        }
211:
212:        if ($relation->element_id == 3)
213:        {
214:            $array = array();
215:            $array[0] = $relation->default_value;
216:
217:         $contentValue->encode = json_encode($array);
218:                            $contentValue->text = $relation->default_value;
219:        }
220:
221:        if ($relation->element_id == 4)
222:        {
223:            $url = $relation->default_value;
224:
225:            $previewClient = new Client($url);
226:
```

```
227:            // Get previews from all available parsers
228:            $previews = $previewClient->getPreviews();
229:
230:            // Get a preview from specific parser
231:                            $preview = $previewClient->getPreview('general');
232:
233:            // Convert output to array
234:            $preview = $preview->toArray();
235:
236:            if (!empty($preview['title']))
237:            {
238:                $title = $preview['title'];
239:                $description = $preview['description'];
240:
241:                $parse = parse_url($url);
242:                $host = $parse['host'];
243:
244:                $array = array();
245:                $array['title'] = $title;
246:                $array['description'] = $description;
247:                $array['url'] = $url;
248:                $array['host'] = $host;
249:
```

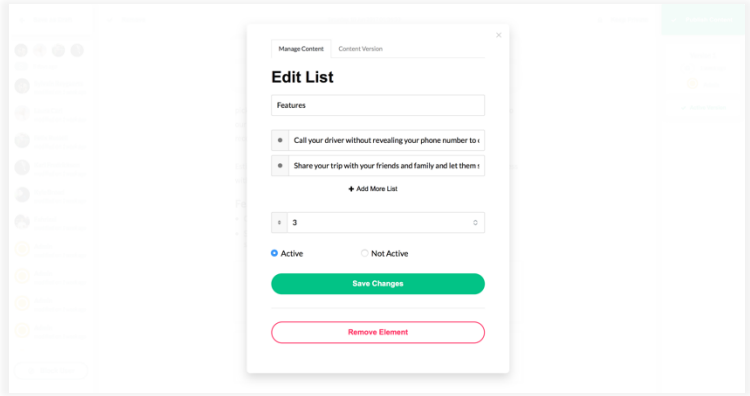
```
250:                                $contentValue->encode =
json_encode($array);
251:            }
252:
253:                            $contentValue->text = $relation-
>default_value;
254:        }
255:
256:        if ($relation->element_id == 5)
257:        {
258:            $contentValue->linked_content = $relation-
>default_value;
259:        }
260:
261:        $contentValue->save();
262:
263:        $contentRelation = new ContentRelation;
264:        $contentRelation->user_id = Auth::user()->id;
265:        $contentRelation->name = $relation->name;
266:          $contentRelation->default_value = $relation-
>default_value;
267:         $contentRelation->value_id = $contentValue-
>id;
268:                $contentRelation->required = $relation-
>required;
269:        $contentRelation->active = $relation->active;
```

```
270:            $contentRelation->order_number = $relation->order_number;
271:              $contentRelation->element_id = $relation->element_id;
272:          $contentRelation->content_id = $contentId;
273:        $contentRelation->version_id = $contentVersion->id;
274:          $contentRelation->save();
275:
276:        $contentValueUpdate = ContentValue::where('id', $contentValue->id)
277:              ->update(['relation_id' => $contentRelation->id]);
278:      }
279:
280:      DB::commit();
281:
282:      return redirect('content/'.$contentId);
283:   }
```

Gambar 5.14 Kode Program Add Content

5.3.6. Implementasi Fitur *Content Element*

Fitur *Content Element* memungkinkan untuk melakukan Perubahan Data / Informasi pada Struktur Data atau Struktur Katalog Produk Perangkat Lunak yang ada pada Konten seperti *Text, Image, List, Link* dan *Inlink* yang dapat dilihat pada Gambar 5.15 Implementasi Fitur Content Element.



Gambar 5.15 Implementasi Fitur Content Element

Seperti tampak pada Gambar 5.16 Kode Program Content Element bahwasannya setiap perubahan yang dilakukan pada *Element* dalam sebuah Konten akan menciptakan sebuah Versi Baru seperti yang dapat dilihat setelah masukan dari pengguna di validasi, pada *function edit* tepatnya dimulai dari baris 675 sampai 788 akan dilakukan pendefinisian data pada aplikasi berdasarkan Jenis Data yang kemudian dibuatlah sebuah Versi Baru yang akan berelasi dan menjadi versi aktif pada *schema* saat ini.

File: app/Http/Controllers/ContentController.php

```
641:    protected function validatorEdit(array $data)
642:    {
643:        return Validator::make($data, [
644:            'content_id' => 'required|integer',
645:            'relation_id' => 'required|integer',
646:            'element_id' => 'required|integer',
647:            'active' => 'required|integer',
```

```
648:             'name' => 'required|max:255',
649:             'order_number' => 'required|integer',
650:         ]);
651:     }
652:
653:     public function edit(Request $request)
654:     {
655:         $validator = $this->validatorEdit($request->all());
656:
657:         if ($validator->fails())
658:         {
659:             $this->throwValidationException(
660:                 $request, $validator
661:             );
662:         }
663:
664:         $data = $request->all();
665:
666:         DB::beginTransaction();
667:
668:         $contentId = $data['content_id'];
669:         $elementId = $data['element_id'];
670:         $relationId = $data['relation_id'];
671:
```

```
672:                                $contentRelationCheck =
ContentRelation::where('id', $relationId)
673:            ->first();
674:
675:        if (empty($data['value']))
676:        {
677:            $defaultValue = $contentRelationCheck->value-
>text; // Text Value
678:        }
679:
680:        if (!empty($data['value']))
681:        {
682:            $defaultValue = $data['value']; // Text Value
683:        }
684:
685:        if ($elementId == 2) // Image Value
686:        {
687:            if (empty($data['value']))
688:            {
689:                    $defaultValue = $contentRelationCheck-
>value->file; // Text Value
690:            }
691:
692:            if (!empty($data['value']))
693:            {
```

```
694:            $validator = Validator::make($request->all(),
[
695:                                                'value' =>
'required|image|mimes:jpeg,png,jpg,gif|max:2048'
696:            ]);
697:
698:            if ($validator->fails())
699:            {
700:              return redirect('content/'.$contentId)->with([
701:                  'warning' => 'Terjadi Kesalahan!! Ukuran
Foto Maksimal : 2MB',
702:                ]);
703:            }
704:
705:                        $imageName = md5(Auth::user()-
>id.Auth::user()->username.$request->value-
>getClientOriginalName()).'.'.$request->value-
>getClientOriginalExtension();
706:                                        $request->value-
>move(public_path('uploads/image/'), $imageName);
707:
708:            $defaultValue = $imageName;
709:          }
710:
711:       }
712:
```

```
713:                                    $contentRelation =
ContentRelation::find($relationId);
714:        $contentRelation->name = $data['name'];
715:        $contentRelation->active = $data['active'];
716:                        $contentRelation->order_number =
$data['order_number'];
717:        $contentRelation->save();
718:
719:                                    $contentValueCheck =
ContentValue::where('relation_id', $relationId)
720:            ->first();
721:
722:        if (!empty($data['value']))
723:        {
724:            $contentValue = new ContentValue;
725:            $contentValue->user_id = Auth::user()->id;
726:
727:            if ($elementId == 1)
728:            {
729:                $contentValue->text = $defaultValue;
730:            }
731:
732:            if ($elementId == 2)
733:            {
734:                $contentValue->file = $defaultValue;
```

```
735:            }
736:
737:            if ($elementId == 3)
738:            {
739:                                        $contentValue->encode =
json_encode($defaultValue);
740:            }
741:
742:            if ($elementId == 4)
743:            {
744:                $url = $defaultValue;
745:
746:                $previewClient = new Client($url);
747:
748:                // Get previews from all available parsers
749:                $previews = $previewClient->getPreviews();
750:
751:                // Get a preview from specific parser
752:                                    $preview = $previewClient-
>getPreview('general');
753:
754:                // Convert output to array
755:                $preview = $preview->toArray();
756:
757:                if (!empty($preview['title']))
```

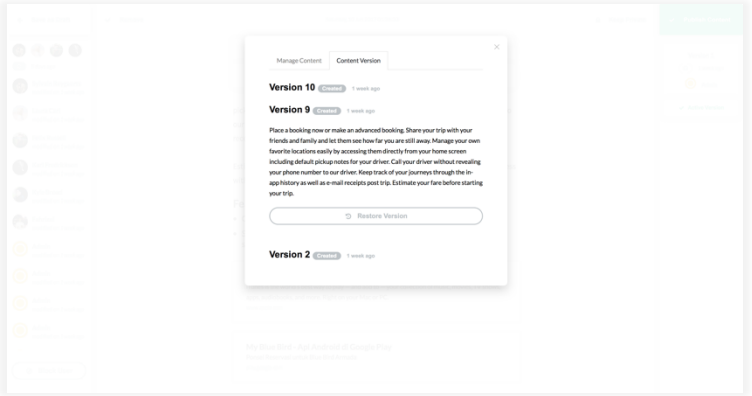
```
758:                {
759:                    $title = $preview['title'];
760:                    $description = $preview['description'];
761:
762:                    $parse = parse_url($url);
763:                    $host = $parse['host'];
764:
765:                    $array = array();
766:                    $array['title'] = $title;
767:                    $array['description'] = $description;
768:                    $array['url'] = $url;
769:                    $array['host'] = $host;
770:
771:                                    $contentValue->encode  =
json_encode($array);
772:                }
773:
774:                $contentValue->text = $defaultValue;
775:            }
776:
777:            if ($elementId == 5)
778:            {
779:                                $contentValue->linked_content  =
$defaultValue;
780:            }
```

```
781:
782:            $contentValue->content_id = $contentId;
783:            $contentValue->relation_id = $relationId;
784:            $contentValue->save();
785:
786:                                    $contentRelationUpdate =
ContentRelation::where('id', $relationId)
787:                ->update(['value_id' => $contentValue->id]);
788:        }
789:
790:        DB::commit();
791:
792:        return redirect('content/'.$contentId);
793:    }
```

Gambar 5.16 Kode Program Content Element

5.3.7. Implementasi Fitur *Content Versioning*

Fitur *Content Versioning* memungkinkan untuk melakukan *Restore Version* terkait Konten maupun Informasi yang diterbitkan untuk mengembalikan Konten maupun Informasi yang ada ke versi-versi sebelumnya baik sebelum maupun sesudah perubahan dilakukan seperti tampak pada Gambar 5.17 Implementasi Fitur Content Versioning.



Gambar 5.17 Implementasi Fitur Content Versioning

Dimana ketika salah satu Versi dari Konten tersebut dipilih dari daftar untuk melakukan *Restore Version* maka sistem akan melakukan pembaruan versi dari Versi Konten yang sedang Aktif ke Versi Konten yang dipilih tersebut seperti tampak pada Gambar 5.18 Kode Program Content Versioning.

File: app/Http/Controllers/ContentController.php

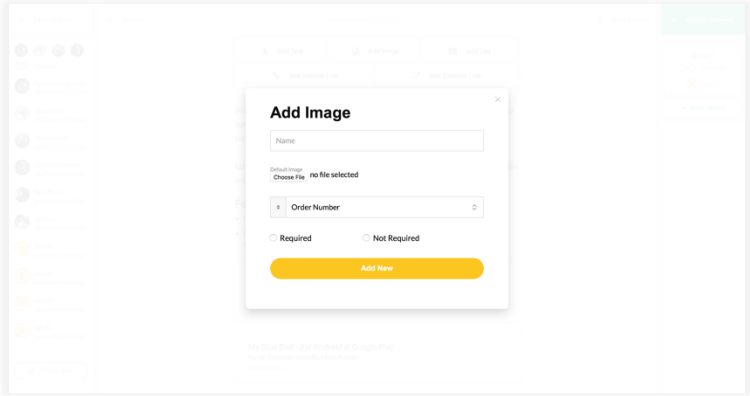
```
364:    public function version(Request $request)
365:    {
366:        $contentId = $request->content_id;
367:
368:        if (!isset($request->version_id))
369:        {
370:            return redirect('/');
371:        }
372:        $versionId = $request->version_id;
373:
374:        $content = Content::where('id', $contentId)
375:            ->update(['version_id' => $versionId]);
376:
377:        return redirect('content/'.$contentId);
378:    }
```

Gambar 5.18 Kode Program Content Versioning

5.3.8. Implementasi Fitur *Content Schema*

5.3.8.1. Implementasi Fitur *Add Content Schema*

Fitur *Add Content Schema* merupakan sebuah fitur yang memungkinkan untuk melakukan penambahan *Element* pada Struktur Data atau Struktur Katalog Produk Perangkat Lunak yang ada seperti *Text, Image, List, Link* dan *Inlink* yang terdapat pada sebuah Konten atau Produk Perangkat Lunak seperti tampak pada Gambar 5.19 Implementasi Fitur Add Content Schema.



Gambar 5.19 Implementasi Fitur Add Content Schema

Kemudian, sistem akan melakukan validasi dari masukan yang ada tersebut untuk membuat sebuah Versi Baru dari *Content Schema* dan membuah sebuah Relasi Baru untuk menghubungkan *Content Schema* dan Versi barunya dengan *Element* yang akan di tambahkan tersebut dan akan dilakukan proses *Relation Cloning* dari Relasi Konten yang lama ke Versi yang baru seperti tampak pada Gambar 5.20 Kode Program Add Content Schema.

File: app/Http/Controllers/ContentController.php

```
413:    protected function validatorAdd(array $data)
414:    {
415:        return Validator::make($data, [
416:            'content_id' => 'required|integer',
417:            'element_id' => 'required|integer',
418:            'active' => 'required|integer',
419:            'name' => 'required|max:255',
420:            'default_value' => 'required',
```

```
421:            'order_number' => 'required|integer',
422:            'required' => 'required|integer',
423:        ]);
424:    }
425:
426:    public function add(Request $request)
427:    {
428:        $validator = $this->validatorAdd($request->all());
429:
430:        if ($validator->fails())
431:        {
432:            $this->throwValidationException(
433:                $request, $validator
434:            );
435:        }
436:
437:        $data = $request->all();
438:
439:        DB::beginTransaction();
440:
441:        $contentId = $data['content_id'];
442:        $elementId = $data['element_id'];
443:
444:            $defaultValue = $data['default_value']; // Text
Value
```

```
445:        if ($elementId == 2) // Image Value
446:        {
447:            $validator = Validator::make($request->all(), [
448:                                                'default_value'   =>
'required|image|mimes:jpeg,png,jpg,gif|max:2048'
449:            ]);
450:
451:            if ($validator->fails())
452:            {
453:                return redirect('content/'.$contentId)->with([
454:                    'warning' => 'Terjadi Kesalahan!! Ukuran
Foto Maksimal : 2MB',
455:                ]);
456:            }
457:
458:                            $imageName = md5(Auth::user()-
>id.Auth::user()->username.$request->default_value-
>getClientOriginalName()).'.'.$request->default_value-
>getClientOriginalExtension();
459:                                        $request->default_value-
>move(public_path('uploads/image/'), $imageName);
460:
461:            $defaultValue = $imageName;
462:        }
463:
464:        $contentCheck = Content::where('id', $contentId)-
>first();
```

```
465:
466:                                $contentVersionCount = ContentVersion::where('content_id', $contentId)
467:            ->count();
468:
469:        $contentVersion = new ContentVersion;
470:        $contentVersion->user_id = Auth::user()->id;
471:                                $contentVersion->number = $contentVersionCount+1;
472:        $contentVersion->content_id = $contentId;
473:        $contentVersion->save();
474:
475:        $versionIdOld = $contentCheck->version_id;
476:        $versionId = $contentVersion->id;
477:
478:        $contentUpdate = Content::where('id', $contentId)
479:            ->update(['version_id' => $versionId]);
480:
481:        $contentValue = new ContentValue;
482:        $contentValue->user_id = Auth::user()->id;
483:
484:        if ($elementId == 1)
485:        {
486:            $contentValue->text = $defaultValue;
487:        }
```

```
488:
489:        if ($elementId == 2)
490:        {
491:            $contentValue->file = $defaultValue;
492:        }
493:
494:        if ($elementId == 3)
495:        {
496:            $array = array();
497:            $array[0] = $defaultValue;
498:
499:            $contentValue->encode = json_encode($array);
500:            $contentValue->text = $defaultValue;
501:        }
502:
503:        if ($elementId == 4)
504:        {
505:            $url = $defaultValue;
506:
507:            $previewClient = new Client($url);
508:
509:            // Get previews from all available parsers
510:            $previews = $previewClient->getPreviews();
511:
```

```
512:          // Get a preview from specific parser
513:                                  $preview = $previewClient-
>getPreview('general');
514:
515:          // Convert output to array
516:          $preview = $preview->toArray();
517:
518:          if (!empty($preview['title']))
519:          {
520:             $title = $preview['title'];
521:             $description = $preview['description'];
522:
523:             $parse = parse_url($url);
524:             $host = $parse['host'];
525:
526:             $array = array();
527:             $array['title'] = $title;
528:             $array['description'] = $description;
529:             $array['url'] = $url;
530:             $array['host'] = $host;
531:
532:           $contentValue->encode = json_encode($array);
533:          }
534:
535:          $contentValue->text = $defaultValue;
```

```
536:        }
537:
538:        if ($elementId == 5)
539:        {
540:          $contentValue->linked_content = $defaultValue;
541:        }
542:
543:        $contentValue->content_id = $contentId;
544:        $contentValue->save();
545:
546:        $contentValueId = $contentValue->id;
547:
548:                                    $contentRelationCopy    =
ContentRelation::where('content_id', $contentId)
549:            ->where('version_id', $versionIdOld)
550:            ->get();
551:
552:        foreach ($contentRelationCopy as $relation)
553:        {
554:            $contentRelation = new ContentRelation;
555:            $contentRelation->user_id = $relation->user_id;
556:            $contentRelation->name = $relation->name;
557:                $contentRelation->default_value = $relation-
>default_value;
```

```
558:                    $contentRelation->value_id = $relation->value_id;
559:                    $contentRelation->required = $relation->required;
560:            $contentRelation->active = $relation->active;
561:              $contentRelation->order_number = $relation->order_number;
562:                  $contentRelation->element_id = $relation->element_id;;
563:                   $contentRelation->content_id = $relation->content_id;
564:            $contentRelation->version_id = $versionId;
565:            $contentRelation->save();
566:        }
567:
568:        $contentRelation = new ContentRelation;
569:        $contentRelation->user_id = Auth::user()->id;
570:        $contentRelation->name = $data['name'];
571:        $contentRelation->default_value = $defaultValue;
572:        $contentRelation->required = $data['required'];
573:        $contentRelation->active = $data['active'];
574:                            $contentRelation->order_number = $data['order_number'];
575:                               $contentRelation->element_id = $data['element_id'];
576:       $contentRelation->content_id = $data['content_id'];
577:        $contentRelation->version_id = $versionId;
```

```
578:        $contentRelation->value_id = $contentValueId;
579:        $contentRelation->save();
580:
581:        $contentValueUpdate = ContentValue::where('id',
$contentValueId)
582:                ->update(['relation_id' => $contentRelation-
>id]);
583:
584:        DB::commit();
585:
586:        return redirect('content/'.$contentId);
587:    }
```

Gambar 5.20 Kode Program Add Content Schema

5.3.8.2. Implementasi Fitur *Remove Content Schema*

Fitur *Remove Content Schema* merupakan sebuah fitur yang berfungsi untuk menghapus *Element* pada sebuah *Content Schema* atau Struktur Data. Seperti tampak pada Gambar 5.21 Kode Program Remove Content Schema akan dibuat sebuah Versi Baru dari *Content Schema* kemudian dilakukan *Relation Cloning* dari Relasi Skema yang lama ke Versi yang baru kecuali Element yang di hapus, hal tersebut perlu dilakukan untuk mendukung Fitur *Schema Versioning*.

File: app/Http/Controllers/ContentController.php

```
589:    public function rem(Request $request)
590:    {
591:        DB::beginTransaction();
592:
593:        $contentId = $request->content_id;
594:        $relationId = $request->relation_id;
595:
596:        $contentCheck = Content::where('id', $contentId)-
>first();
597:
598:                                    $contentVersionCount    =
ContentVersion::where('content_id', $contentId)
599:            ->count();
600:
601:        $contentVersion = new ContentVersion;
602:        $contentVersion->user_id = Auth::user()->id;
603:                                    $contentVersion->number    =
$contentVersionCount+1;
604:        $contentVersion->content_id = $contentId;
605:        $contentVersion->save();
606:
607:        $versionIdOld = $contentCheck->version_id;
608:        $versionId = $contentVersion->id;
609:
```

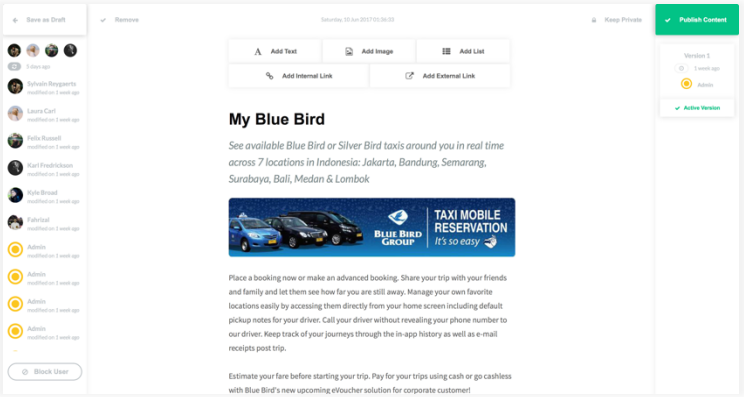
```
610:        $contentUpdate = Content::where('id', $contentId)
611:            ->update(['version_id' => $versionId]);
612:
613:                                    $contentRelationCopy =
ContentRelation::where('content_id', $contentId)
614:            ->where('version_id', $versionIdOld)
615:            ->get();
616:
617:        foreach ($contentRelationCopy as $relation)
618:        {
619:            if($relation->id != $relationId)
620:            {
621:                $contentRelation = new ContentRelation;
622:                        $contentRelation->user_id = $relation-
>user_id;
623:                $contentRelation->name = $relation->name;
624:                $contentRelation->default_value = $relation-
>default_value;
625:                    $contentRelation->value_id = $relation-
>value_id;
626:                    $contentRelation->required = $relation-
>required;
627:                $contentRelation->active = $relation->active;
628:                $contentRelation->order_number = $relation-
>order_number;
629:                    $contentRelation->element_id = $relation-
>element_id;;
```

```
630:                    $contentRelation->content_id = $relation->content_id;
631:                $contentRelation->version_id = $versionId;
632:                $contentRelation->save();
633:            }
634:        }
635:
636:        DB::commit();
637:
638:        return redirect('content/'.$contentId);
639:    }
```

Gambar 5.21 Kode Program Remove Content Schema

5.3.9. Implementasi Fitur *Content Collaboration*

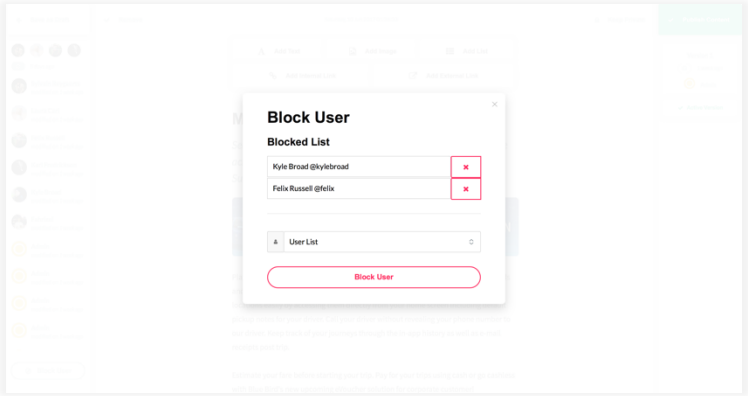
Fitur *Content Collaboration* memungkinkan Para Pengguna untuk saling bergabung dan berkolaborasi untuk meningkatkan kualitas Konten atau Informasi yang ada pada sebuah Konten atau Produk Perangkat Lunak seperti tampak pada Gambar 5.22 Implementasi Fitur Content Collaboration dimana di bagian kiri terdapat informasi terkait siapa saja yang bergabung dan melakukan perubahan dalam Konten tersebut. Dan juga pada bagian pojok kanan atas di samping tombol *Publish Content* terdapat tombol yang berfungsi untuk melakukan pengaturan terhadap status kolaborasi dari Konten tersebut.



Gambar 5.22 Implementasi Fitur Content Collaboration

5.3.10. Implementasi Fitur *User Block*

Fitur *User Block* dapat memungkinkan Pemilik Konten untuk melakukan Pemblokiran pada Pengguna dalam sebuah Konten atau Kolaborasi untuk menghindarkan perubahan atau perusakan Konten atau Informasi yang ada dari Pengguna tersebut seperti tampak pada Gambar 5.23 Implementasi Fitur User Block.



Gambar 5.23 Implementasi Fitur User Block

Dimana terdapat fungsi untuk melakukan *Block User* dan juga *Unblock User* yang digambarkan dalam sebuah tombol dengan icon silang di bagian tengahnya. Seperti tampak pada Gambar 5.24 Kode Program User Block setiap pengguna yang diblokir pada suatu Konten, maka data dirinya akan di simpan pada Informasi Pemblokiran agar saat dia akan masuk ke dalam akun tersebut, sistem akan melakukan penolakan pada pengguna tersebut berdasarkan data diri yang dimiliki oleh pengguna tersebut.

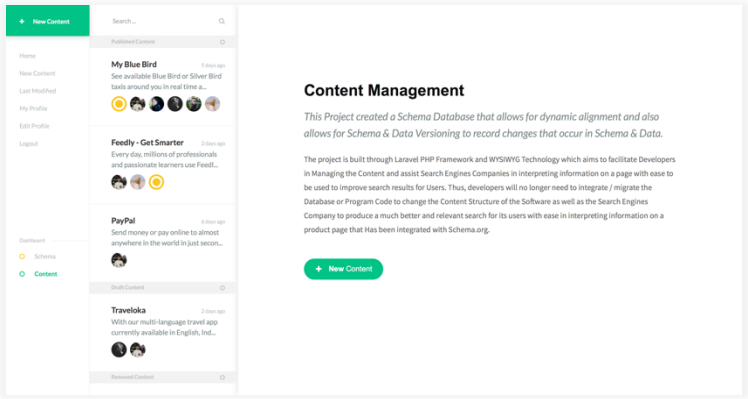
File: app/Http/Controllers/ContentController.php

```
813:    public function block(Request $request)
814:    {
815:        $contentId = $request->content_id;
816:
817:                $username = str_replace('@', '', $request->blocked_id);
818:            $user = User::where('username', $username)->first();
819:        if (empty($user))
820:        {
821:            return redirect('content/'.$contentId);
822:        }
823:
824:        $blockedId = $user->id;
825:
826:        $blocked = new ContentBlock;
827:        $blocked->user_id = Auth::user()->id;
828:        $blocked->content_id = $contentId;
829:        $blocked->blocked_id = $blockedId;
830:        $blocked->save();
831:
832:        return redirect('content/'.$contentId);
833:    }
```

Gambar 5.24 Kode Program User Block

5.3.11. Implementasi Fitur *List Content*

Fitur *List Content* merupakan sebuah fitur yang berfungsi untuk menampilkan seluruh Konten yang pernah di buat berdasarkan Status Penerbitannya seperti *Published, Draft* dan *Removed* seperti tampak pada Gambar 5.25 Implementasi Fitur List Content.



Gambar 5.25 Implementasi Fitur List Content

Dimana Status Penerbitan dari setiap Konten tersebut dibedakan terdiri dari *active* dan *published* status dengan aturan seperti tampak pada Gambar 5.26 Kode Program List Content.

File: app/Http/Controllers/ContentController.php

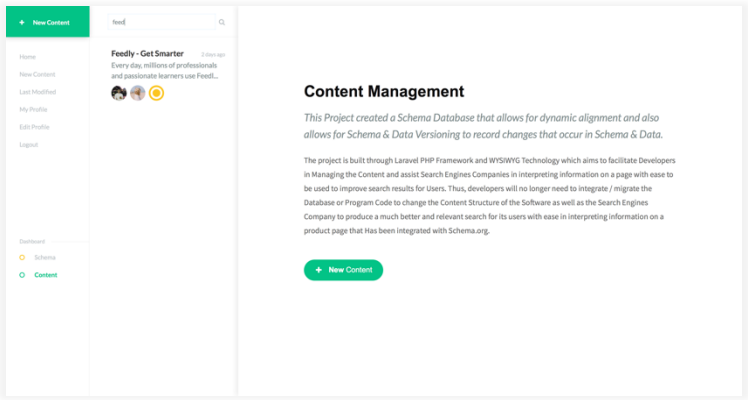
```
24:     public function index()
25:     {
26:         $schemas = Schema::where('active', '1')
27:             ->where('published', '1')
28:             ->get();
29:
30:         $published = Content::where('active', '1')
31:             ->where('published', '1')
32:             ->get();
33:
34:         $draft = Content::where('active', '1')
35:             ->where('published', '0')
36:             ->get();
37:
38:         $removed = Content::where('active', '0')
39:             ->where('published', '0')
40:             ->get();
41:
42:             $count = count($published) + count($draft) +
count($removed);
43:
44:         return view('content.index')->with([
45:             'position' => 'content',
46:             'schemas' => $schemas,
```

```
47:             'published' => $published,
48:             'draft' => $draft,
49:             'removed' => $removed,
50:             'count' => $count,
51:         ]);
52:     }
53:
```

Gambar 5.26 Kode Program List Content

5.3.12. Implementasi Fitur *Search*

Fitur *Search* merupakan sebuah fitur yang berfungsi untuk Melakukan Pencarian terhadap Konten yang ada dan juga membantu Admin untuk dapat Melakukan Pencarian terhadap Skema yang ada seperti tampak pada Gambar 5.27 Implementasi Fitur Search.



Gambar 5.27 Implementasi Fitur Search

Dimana seperti tampak pada Gambar 5.28 Kode Program Search, setiap pencarian yang di lakukan akan di sesuaikan dari posisi atau dari halaman mana pencarian itu dilakukan, apakah dari halaman konten atau halaman skema, atau juga halaman profil untuk pencari sebuah konten secara spesifik dari sebuah pengguna tertentu.

File: app/Http/Controllers/HomeController.php

```
41:    public function search(Request $request)
42:    {
43:        $position = $request->position;
44:        $keyword = $request->keyword;
45:        $results = null;
46:
47:        if (empty($position))
48:        {
49:                    $results = Content::where('title', 'like',
'%'.$keyword.'%')
50:                ->where('active', '1')
51:                ->where('published', '1')
52:                ->orderBy('created_at', 'DESC')
53:                ->get();
54:        }
55:
56:        if (!empty($position))
57:        {
58:            if ($position == 'schema')
59:            {
60:                    $results = Schema::where('name', 'like',
'%'.$keyword.'%')
61:                ->orderBy('created_at', 'DESC')
62:                ->get();
```

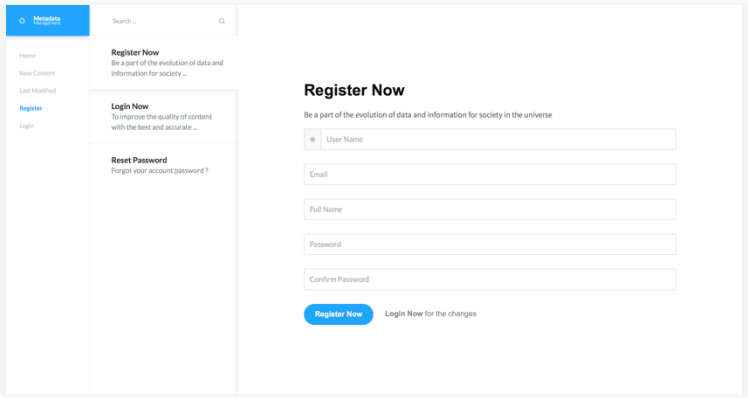
```
63:         }
64:
65:         if ($position == 'content')
66:         {
67:                 $results = Content::where('title', 'like',
'%'.$keyword.'%')
68:             ->orderBy('created_at', 'DESC')
69:             ->get();
70:         }
71:
72:         if ($position == 'profile')
73:         {
74:                 $results = Content::where('title', 'like',
'%'.$keyword.'%')
75:             ->where('user_id', $request->user)
76:             ->where('active', '1')
77:             ->where('published', '1')
78:             ->orderBy('created_at', 'DESC')
79:             ->get();
80:         }
81:     }
82:
83:     return view('public.sidebar.search')->with([
84:         'position' => $position,
85:         'results' => $results,
```

```
86:             'keyword' => $keyword,
87:         ]);
88:
89:     }
```

Gambar 5.28 Kode Program Search

5.3.13. Implementasi Fitur *Register*

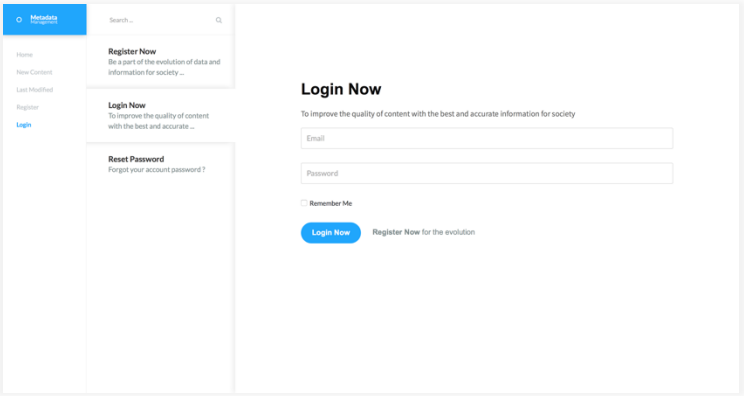
Fitur *Register* merupakan sebuah fitur yang berfungsi untuk mendaftar atau bergabung sebagai *Member* dari aplikasi seperti tampak pada Gambar 5.29 Implementasi Fitur Register.



Gambar 5.29 Implementasi Fitur Register

5.3.14. Implementasi Fitur *Login*

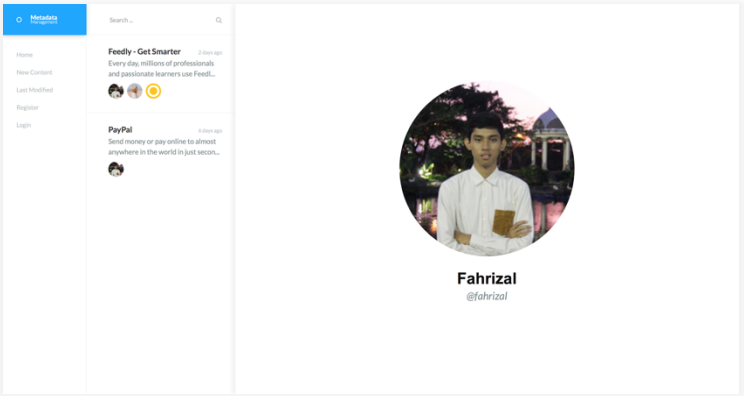
Fitur *Login* merupakan sebuah fitur yang berfungsi untuk masuk ke aplikasi seperti tampak pada Gambar 5.30 Implementasi Fitur Login.



Gambar 5.30 Implementasi Fitur Login

5.3.15. Implementasi Fitur *Profile*

Fitur *Profile* merupakan sebuah fitur yang berfungsi untuk mengetahui Konten apa saja yang telah diterbitkan oleh seorang pengguna maupun Konten apa saja yang dia miliki seperti tampak pada Gambar 5.31 Implementasi Fitur Profile.



Gambar 5.31 Implementasi Fitur Profile

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan terkait hasil dan pembahasan dari proses pengujian perangkat lunak yang telah dilakukan.

6.1. Hasil Pengujian

6.1.1. *Compatibility Testing*

Pada tahap ini akan dilakukan *Compability Testing* yaitu aplikasi akan dijalaan pada skenario pengujian atau *test case* pada beberapa *web browser* populer saat ini diantaranya seperti Safari, Mozilla Firefox dan Google Chrome. Seperti tampak pada Tabel 6.1 Compatibility Testing untuk melakukan *Compatibility Testing* ada beberapa tahapan yang harus dilalui diantaranya:

1. Menentukan Fitur Nomor Berapa yang akan di uji.
2. Membuka *Browser* Safari.
3. Melakukan Pengujian pada Fitur berdasarkan Cara Pengujian yang telah di jelaskan pada kolom Cara Pengujian pada Tabel 6.1 Compatibility Testing.
4. Mencocokkan Hasil Pengujian dengan Hasil yang Diharapkan sesuai dengan apa yang telah di jelaskan pada kolom Hasil yang Diharapkan pada Tabel 6.1 Compatibility Testing.
5. Mencatat Status pengujian pada Tabel 6.1 Compatibility Testing. Jika Hasil Pengujian telah sesuai yang diharapkan maka status pada tabel Hasil Pengujian sesuai dengan Nama *Browser* dilakukannya pengujian akan bertuliskan OK, jika tidak sesuai maka akan dilakukan perbaikan pada Fitur tersebut kemudian dilakukan pengujian ulang dari tahap 1 hingga 5.
6. Membuka *Browser* Firefox, melakukan pengujian berdasarkan tahap 1-5.

7. Membuka *Browser* Chrome, melakukan pengujian berdasarkan tahap 1-5.

Tabel 6.1 Compatibility Testing

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Safari	Firefox	Chrome
1	*Add Schema*	Menekan tombol *New Schema*	Muncul *Modal Box* dengan sebuah *Form* dan Tombol Kuning	OK	OK	OK
2	*Schema Element*	Menekan tombol *Add* pada sebuah *Element*	Muncul *Modal Box* dengan sebuah *Form* dan Tombol Kuning	OK	OK	OK
3	*Schema Versioning*	Melakukan *Scroll* ke arah atas dan bawah Daftar Skema	Versi Skema naik dan turun mengikuti *Scroll* yang dilakukan	OK	OK	OK
4	*List Schema*	Memilih menu *Schema* dengan lingkaran Kuning pada *Sidebar*	Muncul Daftar *Schema* berdasarkan Status Penerbitan	OK	OK	OK
5	*Add Content*	Menekan tombol *New Content*	Muncul *Modal Box* dengan sebuah	OK	OK	OK

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Safari	Firefox	Chrome
			Form dan Tombol Hijau			
6	*Content Element*	Menekan tombol *Add* pada sebuah *Element*	Muncul *Modal Box* dengan sebuah *Form* dan Tombol Kuning	OK	OK	OK
7	*Content Versioning*	Menekan salah satu *Element* kemudian berpindah ke *Tab Content Version*	Muncul Daftar Versi Konten yang tersedia	OK	OK	OK
8	*Content Schema*	Melakukan *Scroll* ke arah atas dan bawah Daftar *Content Schema*	Versi *Content Schema* naik dan turun mengikuti *Scroll* yang dilakukan	OK	OK	OK
9	*Content Collaboration*	Melakukan *Scroll* ke arah atas dan bawah Daftar Pengguna dan Aktivitas	Daftar Pengguna dan Aktifitas muncul mengikuti *Scroll* yang dilakukan	OK	OK	OK
10	*User Block*	Menekan tombol	Muncul *Modal Box*	OK	OK	OK

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Safari	Firefox	Chrome
		Block User Abu-abu	dengan sebuah *Form* dan Tombol Merah			
11	*List Content*	Memilih menu *Content* dengan lingkaran Hijau pada *Sidebar*	Muncul Daftar *Content* berdasarkan Status Penerbitan	OK	OK	OK
12	*Search*	Mengetik kata kunci pada Form *Search*	Muncul Daftar Penemuan dari Pencarian	OK	OK	OK
13	*Register*	Memilih menu *Register* pada *Sidebar*	Muncul Form Pendaftaran *Member*	OK	OK	OK
14	*Login*	Memilih menu *Login* pada *Sidebar*	Muncul Form untuk masuk ke aplikasi	OK	OK	OK
15	*Profile*	Menekan Foto Pengguna pada Konten	Muncul Halaman Profil Pengguna dan Koten yang pernah diterbitkan	OK	OK	OK

6.1.2. *Black Box Testing*

Selain itu juga dilakukan *Black Box Testing* untuk memastikan bahwa semua fungsi dan interaksi yang ada pada aplikasi telah berjalan dengan baik dan sesuai. Seperti tampak pada Tabel 6.2 Black Box Testing untuk melakukan *Black Box Testing* pengujian ini akan dibagi menjadi 3 Bagian berdasarkan *Role* / Peran dari Pengguna pada Aplikasi diantaranya pada Jenis Pengguna *Public* akan melakuka pengujian pada Fitur Nomor 12 sampai 15, Jenis Pengguna *Member* akan melakuka pengujian pada Fitur Nomor 5 sampai 15 dan Jenis Pengguna *Admin* akan melakuka pengujian pada Fitur Nomor 1 sampai 15. Adapun beberapa tahapan yang harus dilalui pada setiap pengujian diantaranya:

1. Menentukan Fitur Nomor Berapa yang akan di uji.
2. Melakukan Pengujian pada Jenis Pengguna yang memiliki Akses pada Fitur tersebut.
3. Membuka *Browser* dan Masuk pada Aplikasi atau *Login* jika jenis pengguna yang di uji adalah *Member* atau *Admin*.
4. Melakukan Pengujian pada Fitur berdasarkan Cara Pengujian yang telah di jelaskan pada kolom Cara Pengujian pada Tabel 6.2 Black Box Testing.
5. Mencocokkan Hasil Pengujian dengan Hasil yang Diharapkan sesuai dengan apa yang telah di jelaskan pada kolom Hasil yang Diharapkan pada Tabel 6.2 Black Box Testing.
6. Mencatat Status pengujian pada Tabel 6.2 Black Box Testing. Jika Hasil Pengujian telah sesuai yang diharapkan maka status pada tabel Hasil Pengujian sesuai dengan Nama Jenis Pengguna dilakukannya pengujian akan bertuliskan OK, jika tidak sesuai maka akan dilakukan perbaikan pada Fitur tersebut kemudian dilakukan pengujian ulang dari tahap 1 hingga 5.

Tabel 6.2 Black Box Testing

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Public	***Member***	***Admin***
1	*Add Schema*	Menekan tombol *New Schema* kemudian mengisi *Form* dan melakukan *Submit*	Muncul Skema yang telah dibuat berdasarkan Informasi pada *Form*	-	-	OK
2	*Schema Element*	Menekan tombol *Add* pada sebuah *Element* kemudian mengisi *Form* dan melakukan *Submit*	*Element* baru muncul pada Lembar Kerja berdasarkan Informasi pada *Form*	-	-	OK
3	*Schema Versioning*	Menekan tombol *Restore Version* pada Daftar Skema	*Element* pada Lembar Kerja berubah berdasarkan versi *Element* pilihan	-	-	OK
4	*List Schema*	Memilih menu *Schema* dengan lingkaran Kuning pada *Sidebar* dan	Muncul Lembar Kerja dan Informasi Skema berdasarkan yang telah dipilih	-	-	OK

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Public	***Member***	***Admin***
		menekan salah satu Skema				
5	*Add Content*	Menekan tombol *New Content* kemudian mengisi *Form* dan melakukan *Submit*	Muncul Konten yang telah dibuat berdasarkan Informasi dan Referensi Skema pada *Form*	-	OK	OK
6	*Content Element*	Menekan tombol *Add* pada sebuah *Element* kemudian mengisi *Form* dan melakukan *Submit*	*Element* baru muncul pada Lembar Kerja berdasarkan Informasi pada *Form*	-	OK	OK
7	*Content Versioning*	Menekan salah satu *Element* kemudian berpindah ke *Tab Content Version* dan menekan tombol *Restore Version*	Informasi Konten berubah mengikuti Informasi Konten yang telah dipilih untuk dilakukan *Restore*	-	OK	OK

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Public	***Member***	***Admin***
8	*Content Schema*	Melakukan Tambah dan Hapus *Element* pada Konten	*Element* melakukan perubahan berdasarkan aksi yang di lakukan	-	OK	OK
9	*Content Collaboration*	Menekan tombol *Keep Private* dan *Open Collaboration*	Konten melakukan pembatasan akses berdasarkan opsi kolaborasi yang dipilih	-	OK	OK
10	*User Block*	Menekan tombol *Block User* Abu-abu dan memilik Pengguna dan melakukan *Submit*	Pengguna yang telah di Blokir tidak dapat melakukan perubahan pada Konten	-	OK	OK
11	*List Content*	Memilih menu *Content* dengan lingkaran Hijau pada *Sidebar* dan menekan salah satu Konten	Muncul Lembar Kerja dan Informasi Konten berdasarkan yang telah dipilih	-	OK	OK
12	*Search*	Mengetik kata kunci	Muncul Daftar	OK	OK	OK

No	Fitur	Cara Pengujian	Hasil yang Diharapkan	Hasil Pengujian		
				Public	*Member*	*Admin*
		pada Form *Search*	Penemuan dari Pencarian			
13	*Register*	Memilih menu *Register* pada *Sidebar* dan mengisi *Form* lalu melakukan *Submit*	Pengguna menjadi *Member* baru dari aplikasi	OK	OK	OK
14	*Login*	Memilih menu *Login* pada *Sidebar* dan mengisi *Form* lalu melakukan *Submit*	Pengguna masuk ke aplikasi sebagai Member	OK	OK	OK
15	*Profile*	Menekan Foto Pengguna pada Konten	Muncul Halaman Profil Pengguna dan Koten yang pernah diterbitkan	OK	OK	OK

6.2. Pembahasan

Pada bagian ini dilakukan pembahasan dan penyimpulan hasil dari serangkaian pengujian *Compatibility Testing* dan *Black Box Testing* yang telah dilakukan pada perangkat lunak.

- **Compatibility Testing** dilakukan untuk memastikan setiap halaman dapat di akses dengan baik dan setiap *User Interface* pada aplikasi dapat bekerja sebagaimana seharusnya yang telah di rancang, pengujian dilakukan pada beberapa *Web Browser* populer yang ada saat ini sebagai tolak ukur dalam melakukan pengujian diantaranya seperti Safari, Mozilla Firefox dan Google Chrome. Pengujian dilakukan kepada 15 Fitur yang dimilik oleh aplikasi dimana pengujian dilakukan secara runtut dari awal hingga akhir pada setiap *Web Browser* dengan hasil akhir bahwa Semua Halaman dapat di akses pada setiap *Web Browser* dan setiap Aksi atau Interaksi yang dilakukan dapat bekerja pada setiap *Web Browser* dengan baik sesuai dengan apa yang telah dirancang.
- **Black Box Testing** dilakukan untuk untuk menguji ketepatan aplikasi apakah hasil eksekusi dari aplikasi telah sesuai dengan apa yang telah di rancang sebelumnya. Pengujian juga dilakukan untuk menemukan kesalahan-kesalahan pada sistem dan struktur data, hal tersebut dilakukan untuk memastikan setiap fungsional aplikasi dapat berjalan dengan baik. Pengujian aplikasi dilakukan berdasarkan Peran Pengguna pada aplikasi diantaranya *Visitor*, *Member* dan Admin. Setiap peran memiliki batasan atas hak akses Informasi dan Halaman pada aplikasi berdasarkan yang sudah dirancang sebelumnya. Hasil akhir pengujian memperlihatkan bahwasannya dari 15 Fitur yang dimiliki oleh aplikasi telah di uji berdasarkan Peran masing-masing Pengguna dapat berjalan dengan baik tanpa ditemukan kendala atau kesalahan sistem.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari hasil penelitian dan saran untuk pengembangan penelitian di masa depan.

7.1. Kesimpulan

Berdasarkan proses-proses yang telah dilakukan dalam pengerjaan penelitian dengan judul "Pengembangan Fitur Pengelolaan Metadata Katalog Produk Pada Situs Penjualan Perangkat Lunak" yang telah dilakukan, dapat disimpulkan sebagai berikut:

1. Untuk mempermudah Pengelolaan Struktur Data Katalog Produk Pada Situs Penjualan Perangkat Lunak, setiap *Schema* dirancang agar dapat bekerja secara independen sesuai dengan kebutuhan atau konteksnya masing-masing namun tetap dapat tetap terhubung dan berelasi dengan setiap data yang dimiliki dan perubahan yang pernah dilakukan. Antarmuka pada Aplikasi dan *Editor* yang digunakan untuk mengelola Struktur Data Katalog Produk telah dirancang semudah mungkin dengan mendukung kebiasaan umum pengguna layaknya menggunakan *Google Form* atau *Microsoft Word* dengan di dukung oleh *Icon* dan Keterangan pada setiap Fungsi / Aksi yang dapat dilakukan pada Aplikasi maupun Lembar Kerja *Schema / Content Editor*.
2. *Schema & Content* memiliki 3 Entitas utama yang saling terhubung diantaranya Entitas Pertama yang berisi Informasi atau Keterangan terkait *Schema / Content* tersebut, Entitas Kedua menyimpan setiap perubahan yang dilakukan pada setiap *Element* dari

Schema / Content dan Entitas Terkahir berfungsi untuk menghubungkan *Schema / Content* tersebut dengan setiap *Element* mana saja yang sedang aktif saat ini dan *Element* mana saja yang telah berubah atau dapat di aktifkan kembali. Rancangan tersebut dibuat agar setiap kali diperlukan perubahan pada Struktur Data Katalog Produk, Migrasi dan Integrasi Basis Data tidak perlu dilakukan pada Aplikasi karena Rancangan Basis Data pada Aplikasi telah dibuat sedemikian rupa agar dapat mengakomodir setiap perubahan Struktur Data Katalog Produk pada Aplikasi. Pengelolaan *Schema & Content Versioning* dapat dilakukan dengan baik melalui rancangan basis data pada aplikasi dengan melakukan *Cloning Version* pada setiap *Schema / Content* jika terjadi perubahan dan setiap perubahan pada *Element / Value* dapat di kembalikan menjadi sejak saat sebelum atau sesudah perubahan tersebut dilakukan.

3. Dengan dukungan Rancangan Basis Data yang memungkinkan dilakukannya perubahan pada setiap Struktur Data Katalog Produk Perangkat Lunak, Kode Program dapat disesuaikan secara langsung dengan Antar Muka pada Rancangan Fitur berdasarkan Alur yang telah didefinisikan pada Aplikasi. Antarmuka telah dirancang untuk dapat merekam setiap perubahan pada *Schema* dan *Content*, murai dari Perubahan pada *Content Versioning, Schema Versioning* dan *Content Collaboration*. Setiap aksi yang dilakukan pada Antarmuka telah dirancang dan di uji dengan benar agar setiap aksi yang dilakukan dapat melakukan Perubahan, Penambahan, Penghapusan dan *Restore* dengan menampilkan hasilnya secara langsung pada Lembar Kerja Aplikasi.

7.2. Saran

Saran untuk penelitian selanjutnya di masa depan:

1. Saat ini terdapat 5 *Element* pada Struktur Data diantaranya seperti *Text, Image, List, Internal Link* dan *External Link*. Harapannya di masa depan *Element* yang dimiliki oleh aplikasi dapat dikembangkan jauh lebih banyak lagi dengan melakukan berbagai macam integrasi oleh sistem eksternal.
2. Untuk meningkatkan peran pengguna dan tingkat kolaborasi untuk membangun konten pada aplikasi. Harapannya di masa depan dapat dilakukan integrasi dengan berbagai sosial media populer seperti Facebook, Twitter dan Google+ untuk lebih menghidupkan informasi pada aplikasi dan membuka akses yang lebih luas lagi bagi setiap orang untuk bergabung, berbagai dan berkolaborasi.
3. Penerapan *Schema.org* pada aplikasi dapat lebih di tingkatkan dan di maksimalkan perannya dalam meningkatkan performa dan kualitas informasi pada aplikasi untuk dapat di baca oleh Mesin Pencari karena peneranan *Schema.org* pada aplikasi saat ini belum terlalu maksimal.
4. Kualitas *User Interface* pada aplikasi dapat di tingkatkan lagi dengan menyesuaikan Konten yang dibawakan oleh aplikasi untuk meningkatkan kualitas penggunaan aplikasi. Karena *User Interface* yang ada saat ini dirancang untuk kebutuhan penggunaan secara luas agar semua orang dapat menggunakan dan dapat digunakan untuk apapun. Oleh karena itu, melakukan spesifikasi *User Interface* berdasarkan Konten aplikasi dapat memberikan nilai lebih kepada pengguna dalam hal *User Experience*.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Statista 2016 Number of apps available in leading app stores as of July 2016 *Statista*

[2] Islam R, Islam R and Mazumder T 2010 Mobile application and its global impact *Int. J. Eng. Technol.* **10** 72–8

[3] Siegmund N, Kästner C, Rosenmüller M, Heidenreich F, Apel S and Saake G 2009 Bridging the Gap between Variability in Client Application and Database Schema. *BTW* pp 297–306

[4] Schema.org SoftwareApplication

[5] King C R and Freeman R K 2015 Flexible database schema

[6] Parker J R, Singh S, Prickril G and Chan W 2015 Integrated collaborative user interface for a document editor program

[7] Seurig A and Spillecke T 2006 Self-describing editors for browser-based WYSIWYG XML/HTML editors

[8] Altshuler D 2004 Method and system for designing, editing and publishing web page content in a live internet session

[9] Underwood J, Neilson P, Char H, Shing D, Horner P, Underwood M, Slaney D and Evesson G 2004 Method and apparatus for generating and modifying multiple instances of element of a web site

[10] Karger D R, Ostler S and Lee R 2009 The web page as a WYSIWYG end-user customizable database-backed information management application *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (ACM) pp 257–60

[11] Verou L, Zhang A X and Karger D R 2016 Mavo: Creating Interactive Data-Driven Web Applications by Authoring HTML *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (ACM) pp 483–96

[12] Battle S, Wood D, Leigh J and Ruth L 2012 The Callimachus project: RDFa as a web template language *Proceedings of the Third International Conference on Consuming Linked Data-Volume 905* (CEUR-WS. org) pp 1–14

[13] Verou M 2013 *Dabblet: A visual IDE for rapid prototyping of client-side web development*

[14] Kowalzcykowski K, Deutsch A, Ong K W, Papakonstantinou Y, Zhao K K and Petropoulos M 2009 Do-It-Yourself database-driven web applications *Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR'09)* (Citeseer)

[15] Greenberg J 2005 Understanding metadata and metadata schemes *Cat. Classif. Q.* **40** 17–36

[16] Rainardi V 2008 *Building a data warehouse: with examples in SQL Server* (John Wiley & Sons)

[17] Ronallo J 2012 HTML5 Microdata and Schema. org *Code4Lib J.* **16**

[18] Barker P and Campbell L M 2014 What is schema. org *LRMI. Retrieved April* **21** 2015

[19] Elmasri R 2008 *Fundamentals of database systems* (Pearson Education India)

[20] Noy N F and Klein M 2004 Ontology evolution: Not the same as schema evolution *Knowl. Inf. Syst.* **6** 428–40

[21] Zhu N 2003 Data versioning systems

[22] Kastrup D 2002 Revisiting WYSIWYG paradigms for

authoring LATEX *Proceedings of the 2002 Annual Meeting, TUGboat* vol 23

[23] Sahuguet A and Azavant F 1999 Wysiwyg web wrapper factory (w4f)

[24] Bean M 2015 *Laravel 5 Essentials* (Packt Publishing Ltd)

[25] Howcroft D and Carroll J 2000 A proposed methodology for Web development *ECIS 2000 Proc.* 73

[26] Pressman R S 2005 *Software engineering: a practitioner's approach* (Palgrave Macmillan)

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Kota Pasuruan pada tanggal 22 Juli 1995. Merupakan anak kedua dari 2 bersaudara dan telah menempuh pendidikan formal yaitu; SD Negeri Pacarkeling 1 Kabupaten Pasuruan, SMP Negeri 6 Kota Pasuruan, dan SMK Negeri 1 Purwosari Kabupaten Pasuruan. Penulis meneruskan pendidikan di Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun 2013 pasca lulus dari pendidikan SMK dan terdaftar sebagai mahasiswa dengan NRP 5213100173.Selama menjadi mahasiswa, penulis aktif dan selalu tertarik mengikuti berbagai kompetisi dan menjalankan bisnis yang berbau seni dan teknologi.

Pada tahun keempat perkuliahan, penulis sempat melakukan magang di PT. IBM Indonesia pada unit IBM Social Business dan juga menjadi Microsoft Student Partner pada tahun 2017. Pengalaman itu juga yang menjadi inspirasi penulis sehingga penulis mengambil topik penelitian ini. Penulis dapat dihubungi melalui email fahrizal@sisfors.com.